A Project Report On

MATPLOTLIB

Submitted to

Rahul Narain

By

Jaskaran Singh - 2022MCS2055
Harsh Singh Chauhan - 2022MCS2052
Battala Divyateja – 2021MCS2126

Department of Computer Science and Engineering

Indian Institute of Technology, Delhi

Academic Session: 2022-23

Contents

1) SOFTWARE ARCHITECTURE

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.
Matplotlib produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, Python/IPython shells, web application servers, and various graphical user interface toolkits.
(source: https://github.com/matplotlib/matplotlib/blob/main/README.md)

Matplotlib Architecture

(source:https://2021.desosa.nl/projects/matplotlib/posts/essay2-vision-to-architecture/)

There are three different layers in the architecture of the matplotlib which are the following:

- Backend Layer
- Artist layer
- Scripting layer

Backend layer

The Backend layer is responsible for the implementation of the code and handles the different environments the users are using matplotlib in. Whereas Scripting and Artist layers can be considered the “frontend” of matplotlib as the user facing plotting code, the Backend layer does the work of creating the figure. It consists of three components: `FigureCanvas`, `Renderer`, and `Event`. `FigureCanvas` is the canvas the figure renders into, depending on the environment the user is operating matplotlib and their request, `Renderer` is responsible for actually rendering the figure into the canvas, and `Event` handles user inputs, such as clicks.

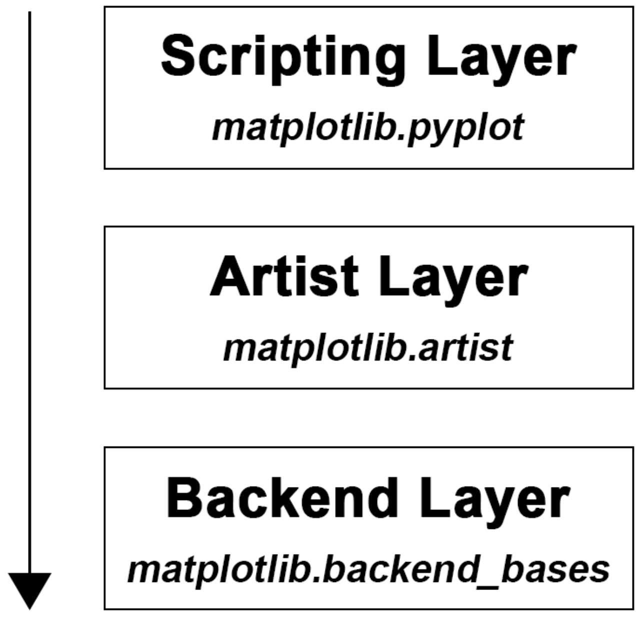
There are two types of backends: user interface backends or interactive backends (used in all interfaces but especially GUI apps), and hardcopy backends that create image files of various formats. Users can set the backend explicitly, but Matplotlib automatically detects a usable backend based on the user’s system and on whether a GUI event loop is running, which is why this layer is rarely used by the regular user.

Artist Layer

The Artist layer is the core of matplotlib, where much of the work is happening. Everything in a matplotlib graph is an Artist instance; the labels, title, ticks, lines, and so on. The OOP API interacts directly with this layer, explicitly creating `Figure` and `Axes` and calling methods on them, which gives users total control over the graph. There are two types of Artist instances: primitive and composite artists. Primitive artists represent the kinds of objects visible in a plot: `Line2D`, `Rectangle`, `Circle`, and `Text`, whereas composite artists are collections of Artist instances, such as `Axis`, `Tick`, `Axes`, and `Figure`.

Scripting layer

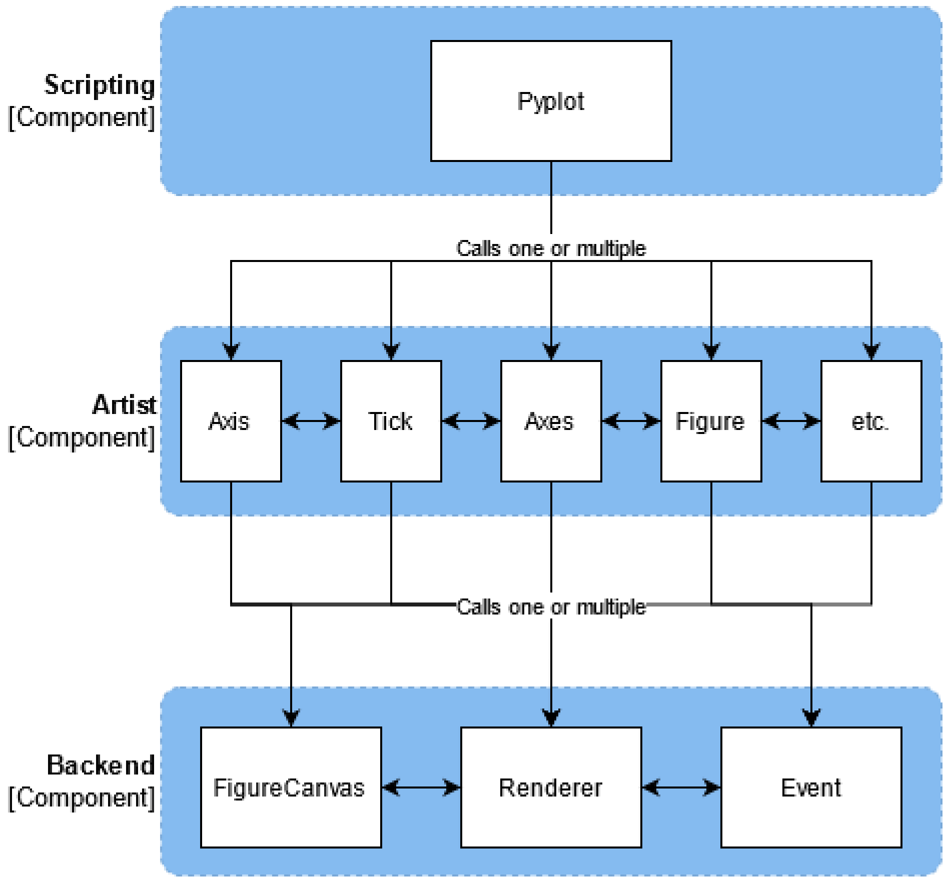
The Scripting layer provides a wrapper around the Artist layer that simplifies the commands by handling most of the repetitive codes used to create `Figure` and `Axes` in Artist layer. However, this simplification results in limited control over the resulting graph, as the Scripting layer already creates the `Figure` and `Axes` objects for the user, and the user can only work on those objects.



(img source: https://junye0798.com/post/plt-xxx-or-ax-xxx-that-is-the-question-in-matplotlib/)

Connectors View

In the last section we have seen the different components of matplotlib, but how do these components connect? Each component can communicate with functions in its layer. The `Event` component can interact with the `Renderer` for example in the backend layer. The lower layers also can act independently from the higher layers while the higher layers depend on the lower layers. The Artist layer will, with help of its classes and functions of the same level, call upon the backend layer to guide the backend layer on what is needed to be drawn. The Artist layer is already all you need to plot most figures, but to make matplotlib *more user-friendly the scripting layer will call the right components of the Artist layer for the task given.*



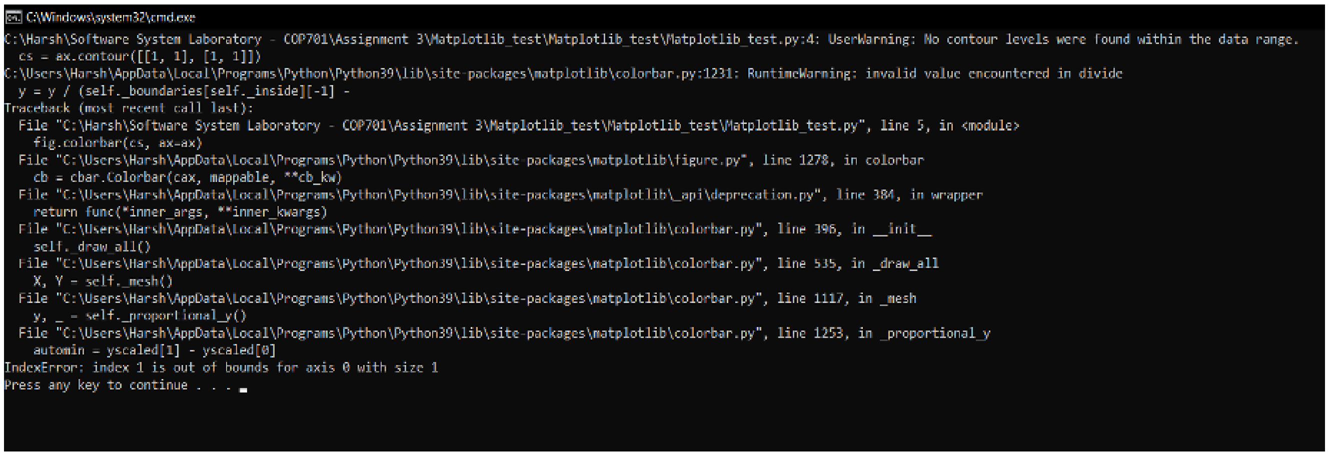
2) Bugs and their Fixes

2.1) Bug : 1

Bug summary

The Axes.contour() function in axes module of matplotlib library is used to plot contours. If you contour a uniform field i.e. ax.contour(z=[[1, 1], [1,1]])) it works fine, although the output is not very interesting. If you add a colorbar to this, you receive an error. Without the colorbar call there is no error.

```python
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
cs = ax.contour([[1, 1], [1, 1]])
fig.colorbar(cs, ax=ax) # Breaking point of the code
plt.show()
```



```
C:\Windows\system32\cmd.exe
C:\Harsh\Software System Laboratory - COP701\Assignment 3\Matplotlib_test\Matplotlib_test\Matplotlib_test.py:4: UserWarning: No contour levels were found within the data range.
  cs = ax.contour([[1, 1], [1, 1]])
C:\Users\Harsh\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\colorbar.py:1231: RuntimeWarning: invalid value encountered in divide
  y = y / (self._boundaries[self._inside][-1] -
Traceback (most recent call last):
  File "C:\Harsh\Software System Laboratory - COP701\Assignment 3\Matplotlib_test\Matplotlib_test\Matplotlib_test.py", line 5, in <module>
    fig.colorbar(cs, ax=ax)
  File "C:\Users\Harsh\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\figure.py", line 1278, in colorbar
    cb = cbar.Colorbar(cax, mappable, **cb_kw)
  File "C:\Users\Harsh\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\_api\deprecation.py", line 384, in wrapper
    return func(*inner_args, **inner_kwargs)
  File "C:\Users\Harsh\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\colorbar.py", line 396, in __init__
    self._draw_all()
  File "C:\Users\Harsh\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\colorbar.py", line 535, in _draw_all
    X, Y = self._mesh()
  File "C:\Users\Harsh\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\colorbar.py", line 1117, in _mesh
    y, _ = self._proportional_y()
  File "C:\Users\Harsh\AppData\Local\Programs\Python\Python39\lib\site-packages\matplotlib\colorbar.py", line 1253, in _proportional_y
    automin = yscaled[1] - yscaled[0]
IndexError: index 1 is out of bounds for axis 0 with size 1
Press any key to continue . . .
```

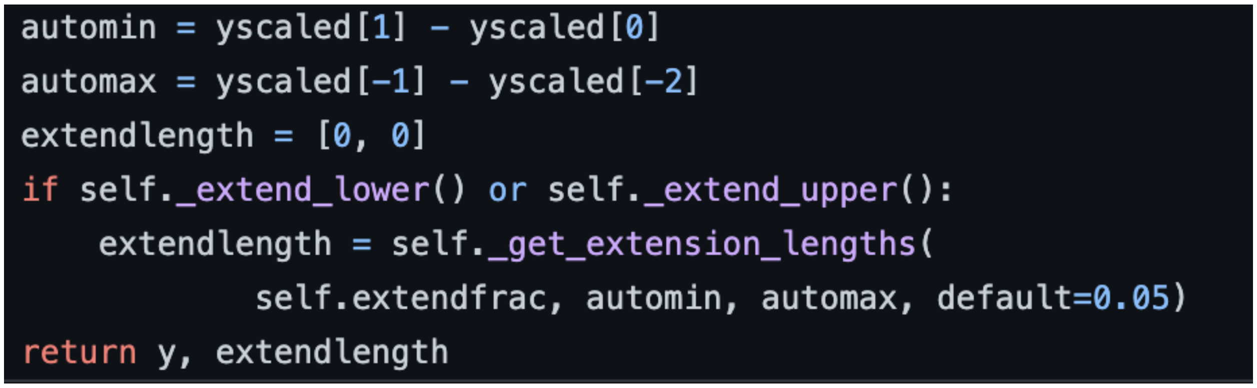
Expected outcome

Colorbar should be displayed as normal without the program getting crashed.

Bug Fix :

The resolution of this issue required changes to be made in the artist layer of architecture of the matplotlib library.

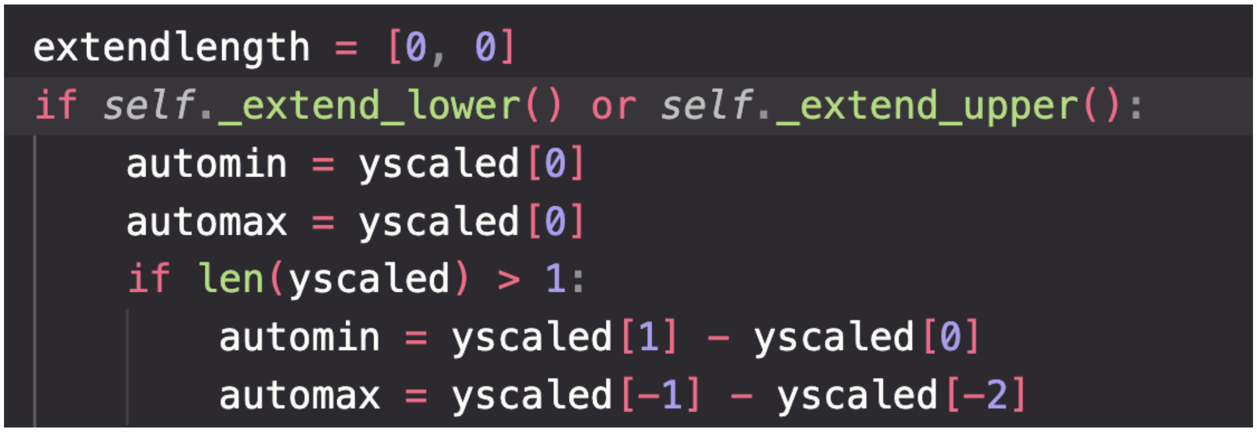
Upon investigating the code we figured that giving the X and Y values as [[1,1],[1,1]] resulted in the yscaled list inside the colorbar to only have a length of 1. Due to the this the following piece of code errored out while being executed :



```
automin = yscaled[1] - yscaled[0]
automax = yscaled[-1] - yscaled[-2]
extendlength = [0, 0]
if self._extend_lower() or self._extend_upper():
    extendlength = self._get_extension_lengths(
            self.extendfrac, automin, automax, default=0.05)
return y, extendlength
```

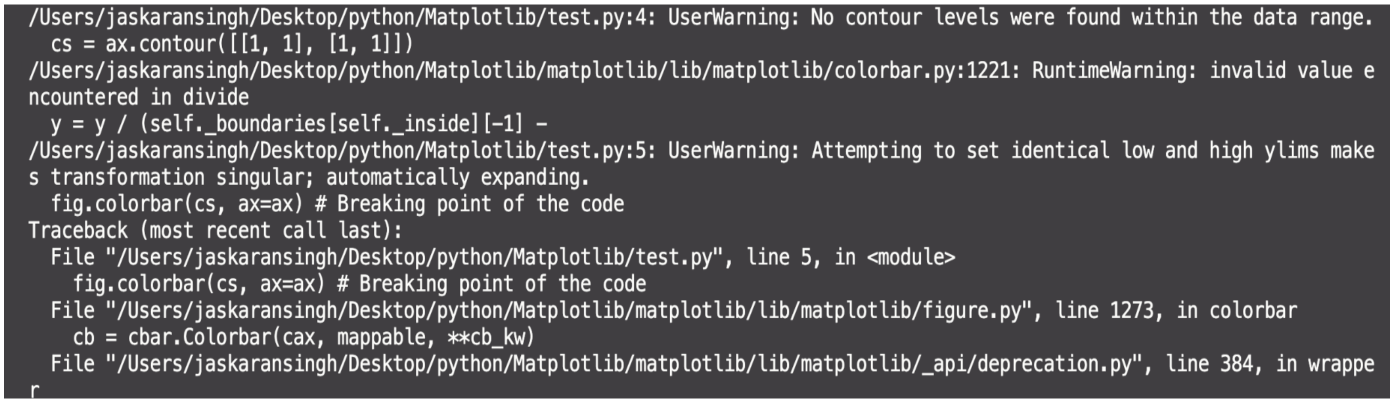
As the length of yscaled list is 1, trying to access the index, yscaled[1] and yscaled[-2] will go out of bounds and give an error.

To fix the above, we moved the above calculations inside the “if” block and laid a check on the length of the yscaled list, and only if the length is greater than 1, then we will calculate the required parameters :

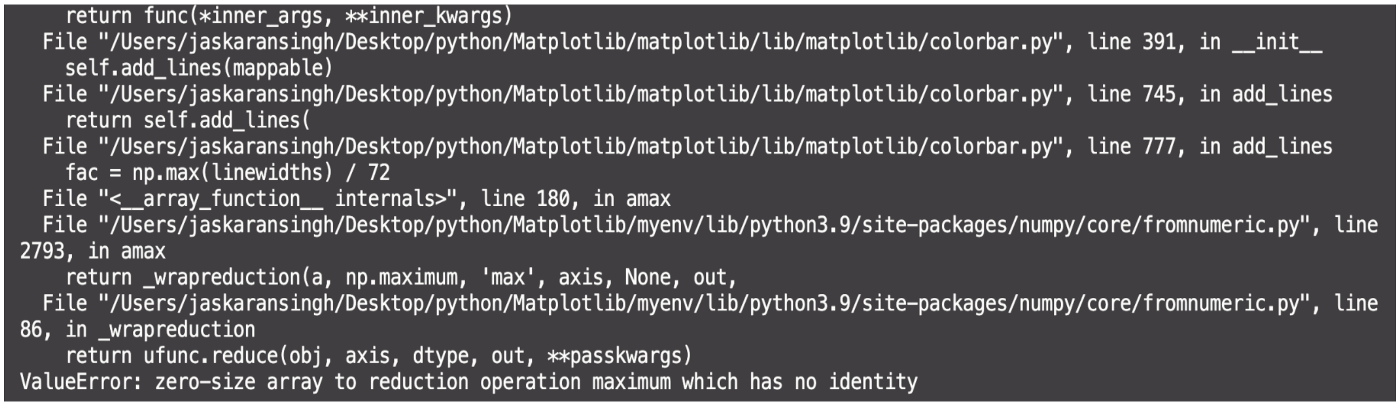


```
extendlength = [0, 0]
if self._extend_lower() or self._extend_upper():
    automin = yscaled[0]
    automax = yscaled[0]
    if len(yscaled) > 1:
        automin = yscaled[1] - yscaled[0]
        automax = yscaled[-1] - yscaled[-2]
```

Upon doing the above, we further ran into another problem :

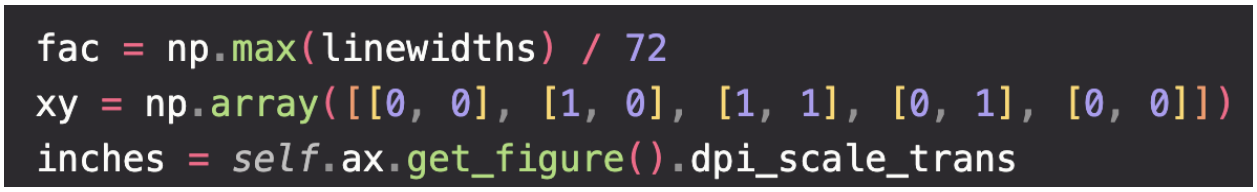


```
/Users/jaskaransingh/Desktop/python/Matplotlib/test.py:4: UserWarning: No contour levels were found within the data range.
  cs = ax.contour([[1, 1], [1, 1]])
/Users/jaskaransingh/Desktop/python/Matplotlib/matplotlib/lib/matplotlib/colorbar.py:1221: RuntimeWarning: invalid value e
ncountered in divide
  y = y / (self._boundaries[self._inside][-1] -
/Users/jaskaransingh/Desktop/python/Matplotlib/test.py:5: UserWarning: Attempting to set identical low and high ylims make
s transformation singular; automatically expanding.
  fig.colorbar(cs, ax=ax) # Breaking point of the code
Traceback (most recent call last):
  File "/Users/jaskaransingh/Desktop/python/Matplotlib/test.py", line 5, in <module>
    fig.colorbar(cs, ax=ax) # Breaking point of the code
  File "/Users/jaskaransingh/Desktop/python/Matplotlib/matplotlib/lib/matplotlib/figure.py", line 1273, in colorbar
    cb = cbar.Colorbar(cax, mappable, **cb_kw)
  File "/Users/jaskaransingh/Desktop/python/Matplotlib/matplotlib/lib/matplotlib/_api/deprecation.py", line 384, in wrappe
r
```



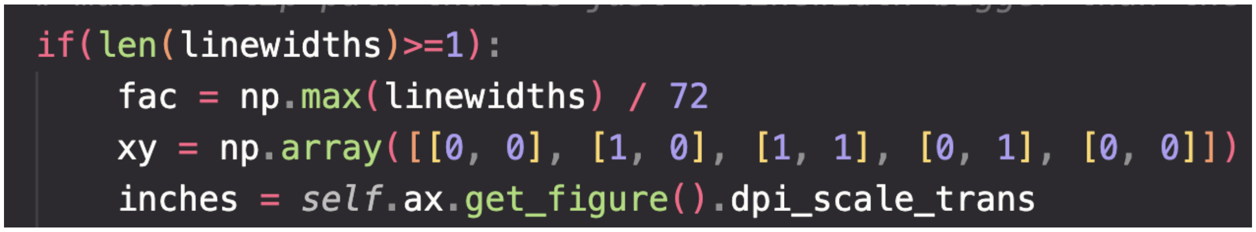
```
    return func(*inner_args, **inner_kwargs)
  File "/Users/jaskaransingh/Desktop/python/Matplotlib/matplotlib/lib/matplotlib/colorbar.py", line 391, in __init__
    self.add_lines(mappable)
  File "/Users/jaskaransingh/Desktop/python/Matplotlib/matplotlib/lib/matplotlib/colorbar.py", line 745, in add_lines
    return self.add_lines(
  File "/Users/jaskaransingh/Desktop/python/Matplotlib/matplotlib/lib/matplotlib/colorbar.py", line 777, in add_lines
    fac = np.max(linewidths) / 72
  File "<__array_function__ internals>", line 180, in amax
  File "/Users/jaskaransingh/Desktop/python/Matplotlib/myenv/lib/python3.9/site-packages/numpy/core/fromnumeric.py", line
2793, in amax
    return _wrapreduction(a, np.maximum, 'max', axis, None, out,
  File "/Users/jaskaransingh/Desktop/python/Matplotlib/myenv/lib/python3.9/site-packages/numpy/core/fromnumeric.py", line
86, in _wrapreduction
    return ufunc.reduce(obj, axis, dtype, out, **passkwargs)
ValueError: zero-size array to reduction operation maximum which has no identity
```

Upon investigating, we found out the problem in this segment of the code :



```
fac = np.max(linewidths) / 72
xy = np.array([[0, 0], [1, 0], [1, 1], [0, 1], [0, 0]])
inches = self.ax.get_figure().dpi_scale_trans
```

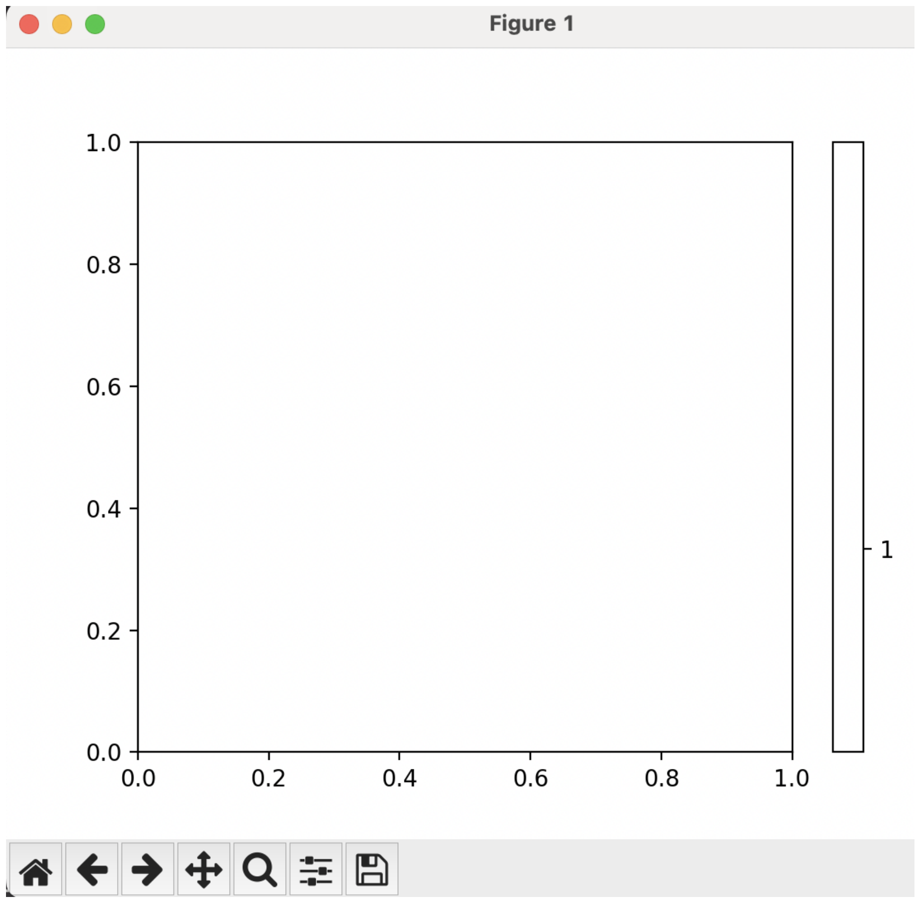
The length of the np.array linewidths was coming to be 0, and the max function being called on the linewidth was crashing the program. Thus, we introduced the following code segment :



```
if(len(linewidths)>=1):
    fac = np.max(linewidths) / 72
    xy = np.array([[0, 0], [1, 0], [1, 1], [0, 1], [0, 0]])
    inches = self.ax.get_figure().dpi_scale_trans
```

This made sure that the max function will only be called when we have a valid length array.

We were able to fix the bug by following the above listed steps and the following output was received :



We can see the color bar appearing on the right side of the graph.

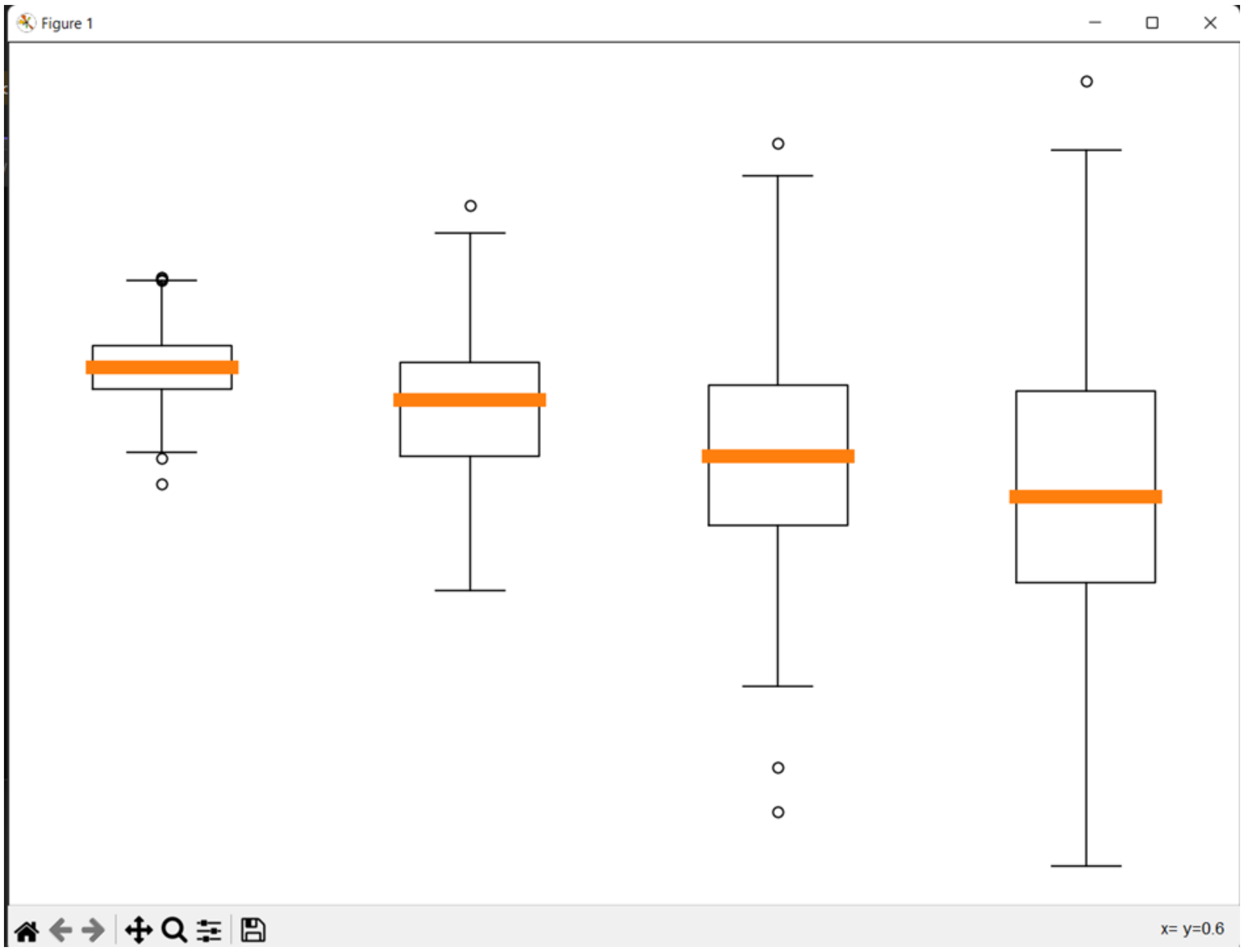
2.2) Bug : 2

Bug summary

When changing the linewidth property of the median line in boxplots, the line will extend beyond the box width.
For example,

```
# Import libraries

import matplotlib.pyplot as plt

import numpy as np
```

```
# Creating dataset

np.random.seed(10)


data_1 = np.random.normal(100, 10, 200)

data_2 = np.random.normal(90, 20, 200)

data_3 = np.random.normal(80, 30, 200)

data_4 = np.random.normal(70, 40, 200)

data = [data_1, data_2, data_3, data_4]


fig = plt.figure(figsize =(10, 7))


# Creating axes instance

ax = fig.add_axes([0, 0, 1, 1])


# Creating plot

bp = ax.boxplot(data, medianprops={"linewidth": 8})


# show plot

plt.show()
```



Expected Outcome:

Changing the linewidth property should only increase the width of the orange lines above and not the length. The way that the lines are coming out of the box is not desirable and we want this line to not extend beyond the box of the boxplot.

Bug Fix :

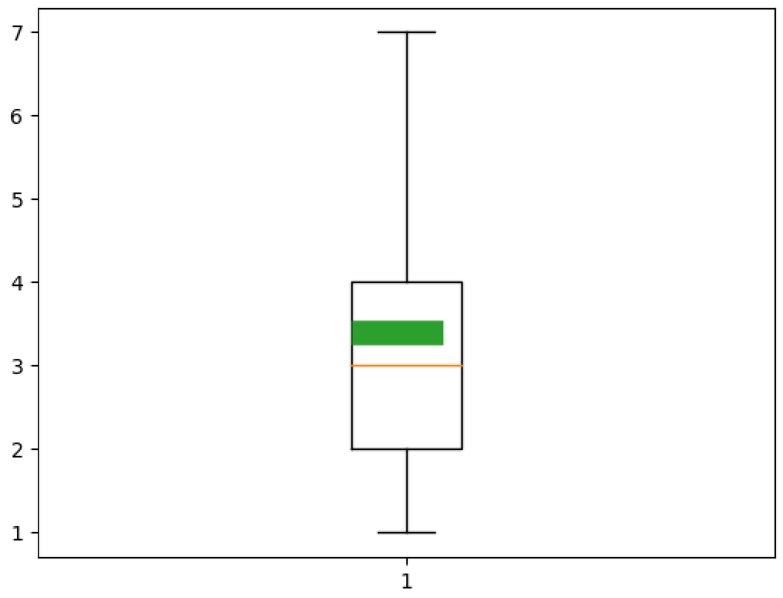
The resolution of this issue required changes to be made in the artist layer of architecture of the matplotlib library.

We observed that the bug mentioned above was only occurring in case we were increasing the line width of the median line of the boxplot. By increasing the linewidth of the mean line, there didn't seem to be the same issue.

Code for reproduction:

```
import matplotlib.pyplot as plt

data = [[1, 2, 3, 4, 7]]

fig, ax = plt.subplots()

ax.boxplot(data,

        showmeans=True,

        meanline=True,

        meanprops={"linewidth": 12})

plt.show()
```

<u>Result</u>



(fig. 1)

So the issue only seems to be with the median line's linewidth property.

Now, in order to solve the issue, we first had to get some idea about matplotlib. Although we had used it before, we never used it extensively enough to know about the fine details that go along with the plots.

There are loads of parameters which can be passed around while making a function call which can be used to finely adjust the plots according to one's need.

So, the first step we took in order to solve the issue was to try and read the discussion that was going on in the thread which stated this bug as an open issue.

It was really very difficult to make any sense of what was being discussed over there as we didn't have an extensive experience with matplotlib before and everything sounded quite new and overwhelming.

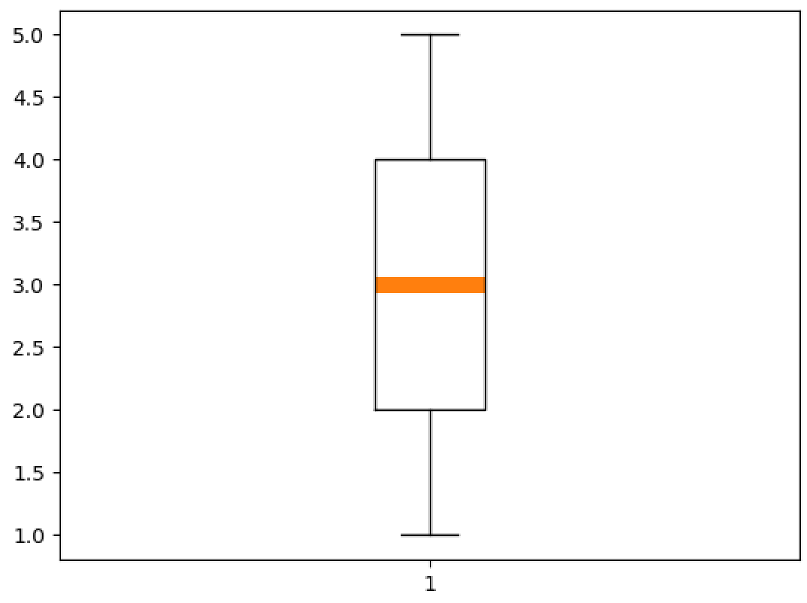
One of the comments said something about the zorder value. So, we had to read about this and how it can affect the plot.

According to https://matplotlib.org/stable/gallery/misc/zorder_demo.html

"The drawing order of artists is determined by their zorder attribute, which is a floating point number. Artists with higher zorder are drawn on top. You can change the order for individual artists by setting their zorder. The default value depends on the type of the Artist."

Another thing which we came across while researching about how to fix the bug is about the solid capstyle of the median line.

We figured that If we set the default capstyle of the medianprops as "butt" the default plot always comes out in a way that the median line stays inside the box and doesn't exceed beyond the box or for that matter, overlap it. (as can be seen in the fig. below)



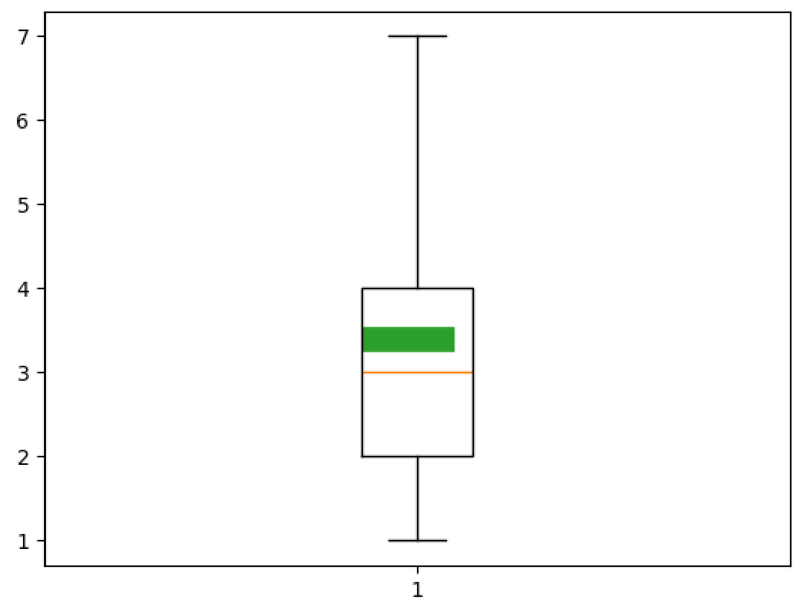
(fig. 2)

Another Bug Found:

While reading about zorder and going through the discussion thread, another slight issue was found which said that in the above plot (fig.1), the mean line and the median line are overlapping with the box's line. We ideally want these lines to be inside the box's line.

We also solved this issue by setting the default value of zorder to 2 which allowed the box plot to be over the mean and median line.

This solved the issue as can be seen below:



If we zoom into the figure above, we will notice that the mean and median lines no longer overlap with the box.

2.3) Bug : 3

Bug summary

There is a ValueError: RGBA sequence should have length 3 or 4 when adding a coloured text legend to a scatter plot using any of the values for the argument labelcolor: linecolor, markerfacecolor, or markeredgecolor.

```
import matplotlib.pyplot as plt

import numpy as np

fig, ax = plt.subplots()

ax.scatter(np.random.rand(500)-0.65, np.random.rand(500), label="First
dataset")
```

ax.scatter(np.random.rand(500)-0.35, np.random.rand(500), label="Second dataset")

ax.legend(labelcolor="linecolor")

fig.savefig("/tmp/test.png")

The following error is received :

```
File
"/Users/jaskaransingh/opt/anaconda3/lib/python3.9/site-packages/matplotlib/backend_bases.py", line 1193, in _on_timer
    ret = func(*args, **kwargs)
  File
"/Users/jaskaransingh/opt/anaconda3/lib/python3.9/site-packages/matplotlib/backends/backend_macosx.py", line 68, in callback_func
    callback()
  File
"/Users/jaskaransingh/opt/anaconda3/lib/python3.9/site-packages/matplotlib/backends/backend_macosx.py", line 88, in _draw_idle
    self.draw()
  File
"/Users/jaskaransingh/opt/anaconda3/lib/python3.9/site-packages/matplotlib/backends/backend_macosx.py", line 50, in draw
    super().draw()
  File
"/Users/jaskaransingh/opt/anaconda3/lib/python3.9/site-packages/matplotlib/backends/backend_agg.py", line 405, in draw
    self.figure.draw(self.renderer)
  File
"/Users/jaskaransingh/opt/anaconda3/lib/python3.9/site-packages/matplotlib/artist.py", line 74, in draw_wrapper
    result = draw(artist, renderer, *args, **kwargs)
  File
"/Users/jaskaransingh/opt/anaconda3/lib/python3.9/site-packages/matplotlib/artist.py", line 51, in draw_wrapper
    return draw(artist, renderer)
  File
"/Users/jaskaransingh/opt/anaconda3/lib/python3.9/site-packages/matplotlib/figure.py", line 3071, in draw
```

```
    mimage._draw_list_compositing_images(
  File
"/Users/jaskaransingh/opt/anaconda3/lib/python3.9/site-packages/matplotlib/imag
e.py", line 131, in _draw_list_compositing_images
    a.draw(renderer)
  File
"/Users/jaskaransingh/opt/anaconda3/lib/python3.9/site-packages/matplotlib/artist
.py", line 51, in draw_wrapper
    return draw(artist, renderer)
  File
"/Users/jaskaransingh/opt/anaconda3/lib/python3.9/site-packages/matplotlib/axes
/_base.py", line 3107, in draw
    mimage._draw_list_compositing_images(
  File
"/Users/jaskaransingh/opt/anaconda3/lib/python3.9/site-packages/matplotlib/imag
e.py", line 131, in _draw_list_compositing_images
    a.draw(renderer)
  File
"/Users/jaskaransingh/opt/anaconda3/lib/python3.9/site-packages/matplotlib/artist
.py", line 51, in draw_wrapper
    return draw(artist, renderer)
  File
"/Users/jaskaransingh/opt/anaconda3/lib/python3.9/site-packages/matplotlib/lege
nd.py", line 649, in draw
    self._legend_box.draw(renderer)
  File
"/Users/jaskaransingh/opt/anaconda3/lib/python3.9/site-packages/matplotlib/offse
tbox.py", line 366, in draw
    c.draw(renderer)
  File
"/Users/jaskaransingh/opt/anaconda3/lib/python3.9/site-packages/matplotlib/offse
tbox.py", line 366, in draw
    c.draw(renderer)
  File
"/Users/jaskaransingh/opt/anaconda3/lib/python3.9/site-packages/matplotlib/offse
tbox.py", line 366, in draw
    c.draw(renderer)
  [Previous line repeated 1 more time]
  File
"/Users/jaskaransingh/opt/anaconda3/lib/python3.9/site-packages/matplotlib/offse
tbox.py", line 814, in draw
    self._text.draw(renderer)
  File
"/Users/jaskaransingh/opt/anaconda3/lib/python3.9/site-packages/matplotlib/artist
.py", line 51, in draw_wrapper
    return draw(artist, renderer)
  File
"/Users/jaskaransingh/opt/anaconda3/lib/python3.9/site-packages/matplotlib/text.
py", line 707, in draw
    gc.set_foreground(self.get_color())
  File
"/Users/jaskaransingh/opt/anaconda3/lib/python3.9/site-packages/matplotlib/back
end_bases.py", line 948, in set_foreground
    self._rgb = colors.to_rgba(fg)
  File
"/Users/jaskaransingh/opt/anaconda3/lib/python3.9/site-packages/matplotlib/color
s.py", line 299, in to_rgba
    rgba = _to_rgba_no_colorcycle(c, alpha)
  File
"/Users/jaskaransingh/opt/anaconda3/lib/python3.9/site-packages/matplotlib/color
s.py", line 383, in _to_rgba_no_colorcycle
    raise ValueError("RGBA sequence should have length 3 or 4")
ValueError: RGBA sequence should have length 3 or 4
```

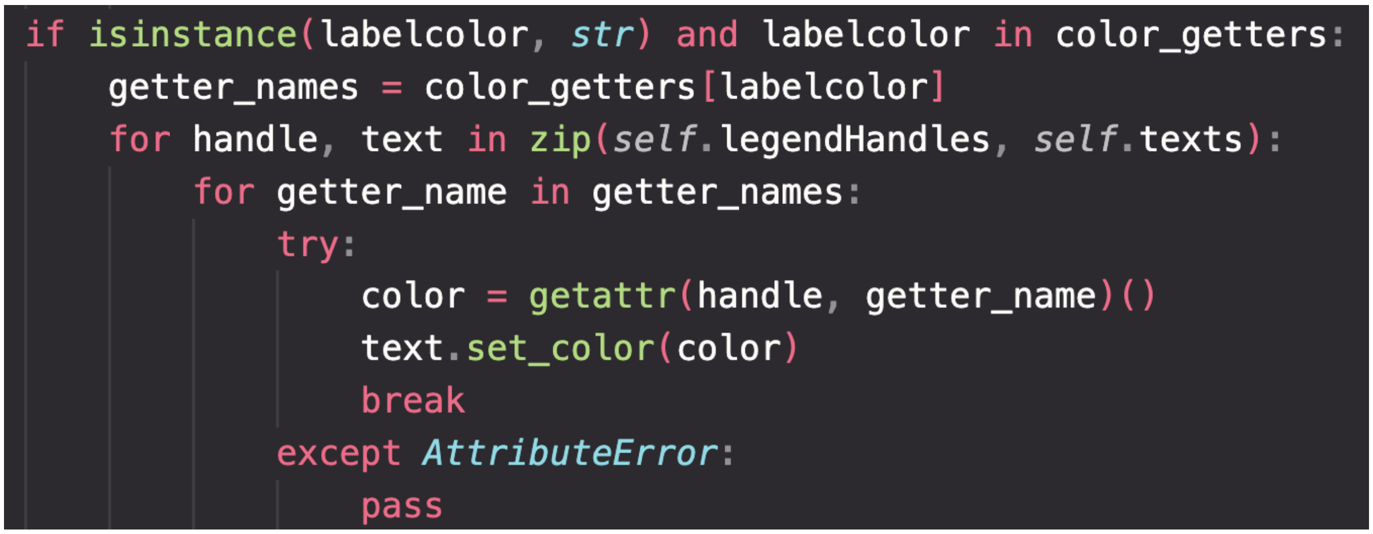
Expected outcome

The legend text should match the color of the points

Bug Fix :

The way scatter plot is defined in Matplotlib, it allows multiple colours to be provided for each marker independently in the scatter plot. If no color is provided then a color at random is selected for all the markets of the scatterplot.
The default behaviour of the legend is that if there is only one color in the scatter plot, the color of the text of the legend will be the same. But if the same scatter plot has multiple colors for it's markers then when we try to define a legend, the program crashes because there is ambiguity while selecting the color of the text that has to be chosen by the legend.
Basically, one would like to detect if color is not a single color value but a list of color values.

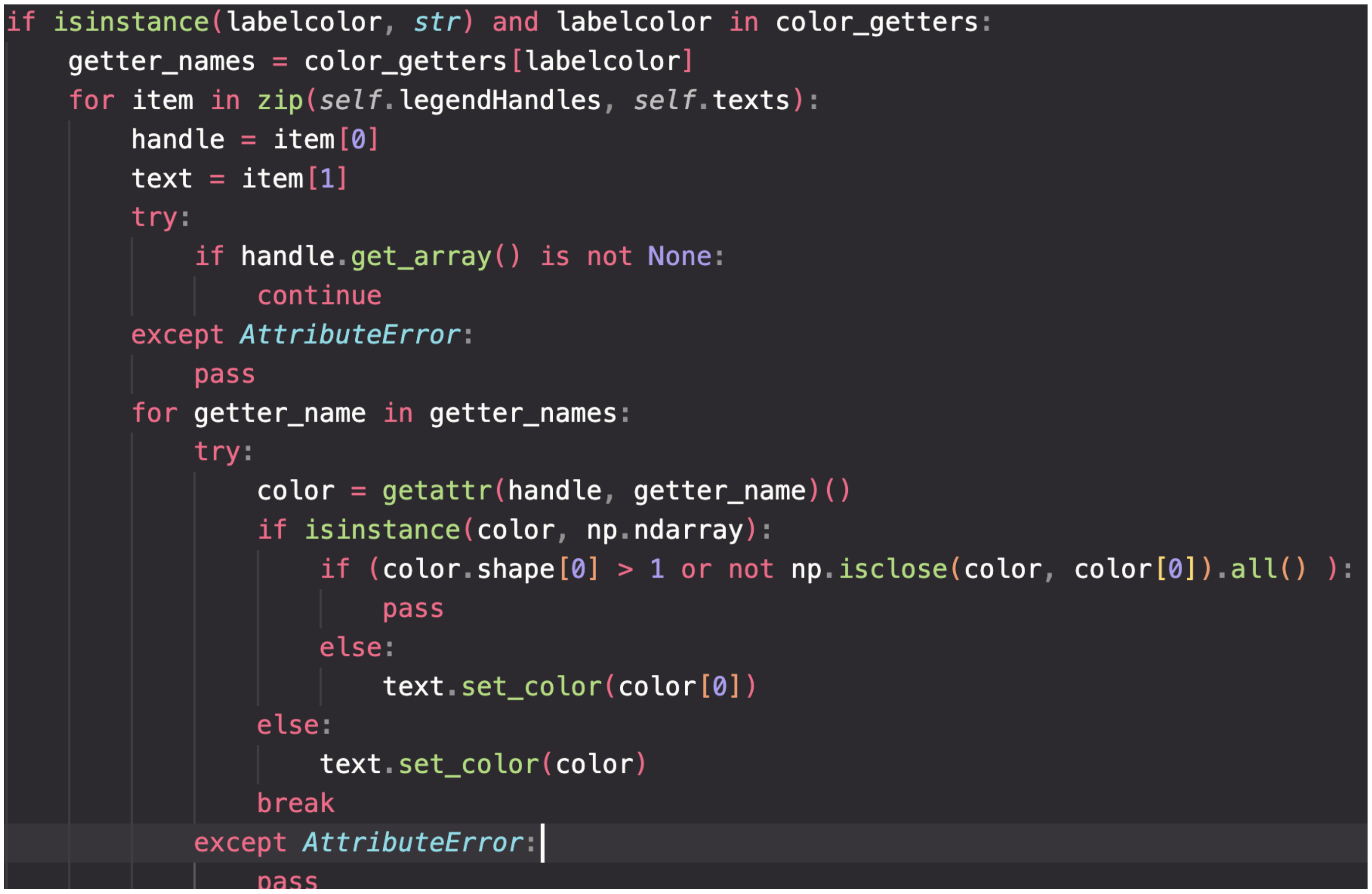
This is the piece of code which was handling the colors that the legend has to take :



```python
if isinstance(labelcolor, str) and labelcolor in color_getters:
    getter_names = color_getters[labelcolor]
    for handle, text in zip(self.legendHandles, self.texts):
        for getter_name in getter_names:
            try:
                color = getattr(handle, getter_name)()
                text.set_color(color)
                break
            except AttributeError:
                pass
```

In the case of a multi-color scatter (either fixed-colors or colormapped) we use the defaulttext color and not apply any artist color. According to the raised issue, setting `labelcolor` to any of the string options did not work with a scatter plot as it is a PathCollection with possible multiple color values. Applying the below code fixes that. Now, if there is a Colormap or a scatter plot with multiple values, then, the legend text color is not changed, but otherwise, the text color is changed to the color of the scatter markers.

The following is introduced to handle the above problem :



```python
if isinstance(labelcolor, str) and labelcolor in color_getters:
    getter_names = color_getters[labelcolor]
    for item in zip(self.legendHandles, self.texts):
        handle = item[0]
        text = item[1]
        try:
            if handle.get_array() is not None:
                continue
        except AttributeError:
            pass
        for getter_name in getter_names:
            try:
                color = getattr(handle, getter_name)()
                if isinstance(color, np.ndarray):
                    if (color.shape[0] > 1 or not np.isclose(color, color[0]).all() ):
                        pass
                    else:
                        text.set_color(color[0])
                else:
                    text.set_color(color)
                break
            except AttributeError:
                pass
```

Whenever a colormap or color array of size more than one is provided, then text.set_color(color) will be called, otherwise we use the default single color of the scatterplot marker.

3)Additional Requirements

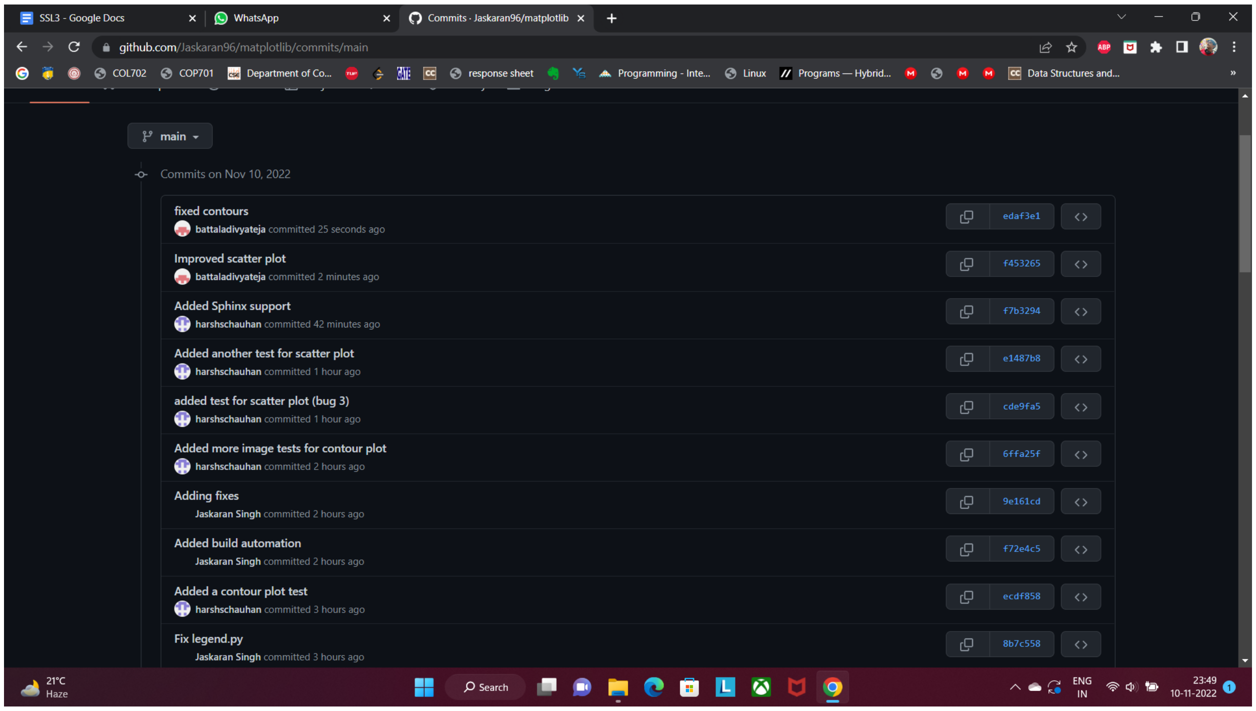
3.1) Version Control – GIT

“Version control, also known as source control, is the practice of tracking and managing changes to software code. Version control systems are software tools that help software teams manage changes to source code over time. As development environments have accelerated, version control systems help software teams work faster and smarter. Version control software keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members.”
(Source - https://www.atlassian.com/git/tutorials/what-is-version-control)

“**Git** is free and open-source software for distributed version control: tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. Its goals include speed, data integrity, and support for distributed, non-linear workflows (thousands of parallel branches running on different systems)”
(Source - https://en.wikipedia.org/wiki/Git)

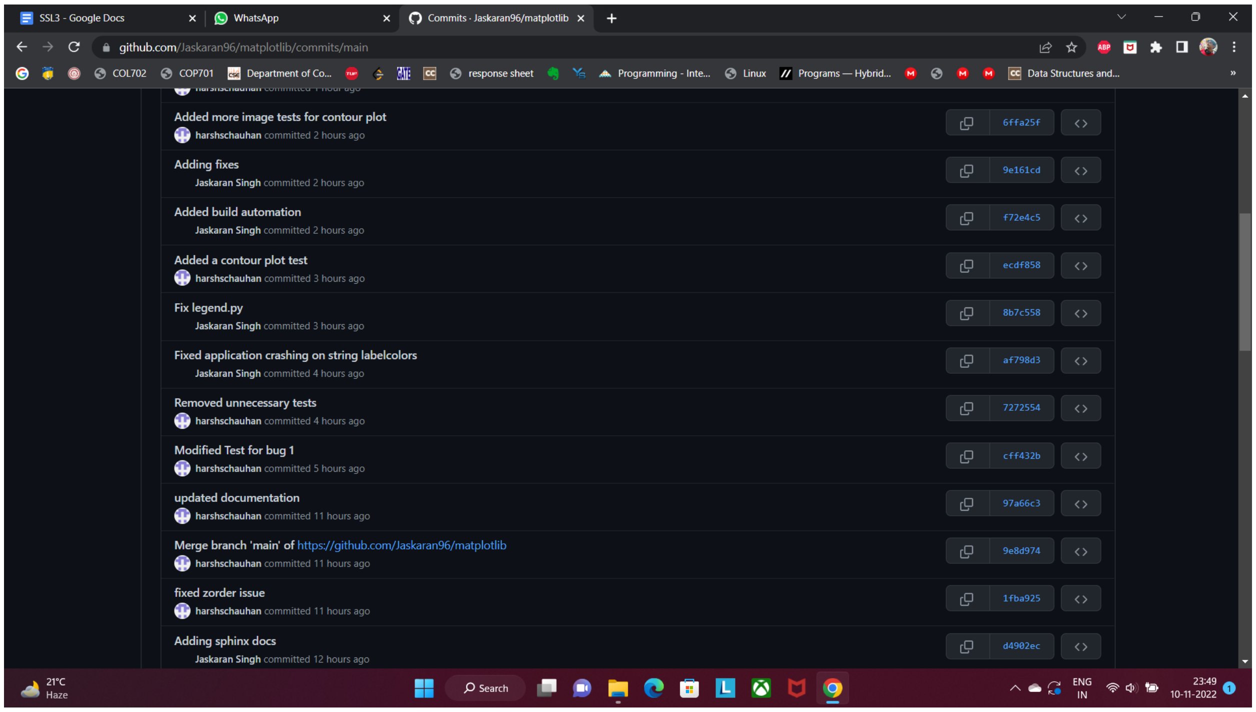
We have leveraged GIT as the version control in our application.

Below is the attached screenshots of our GIT COMMIT HISTORY :

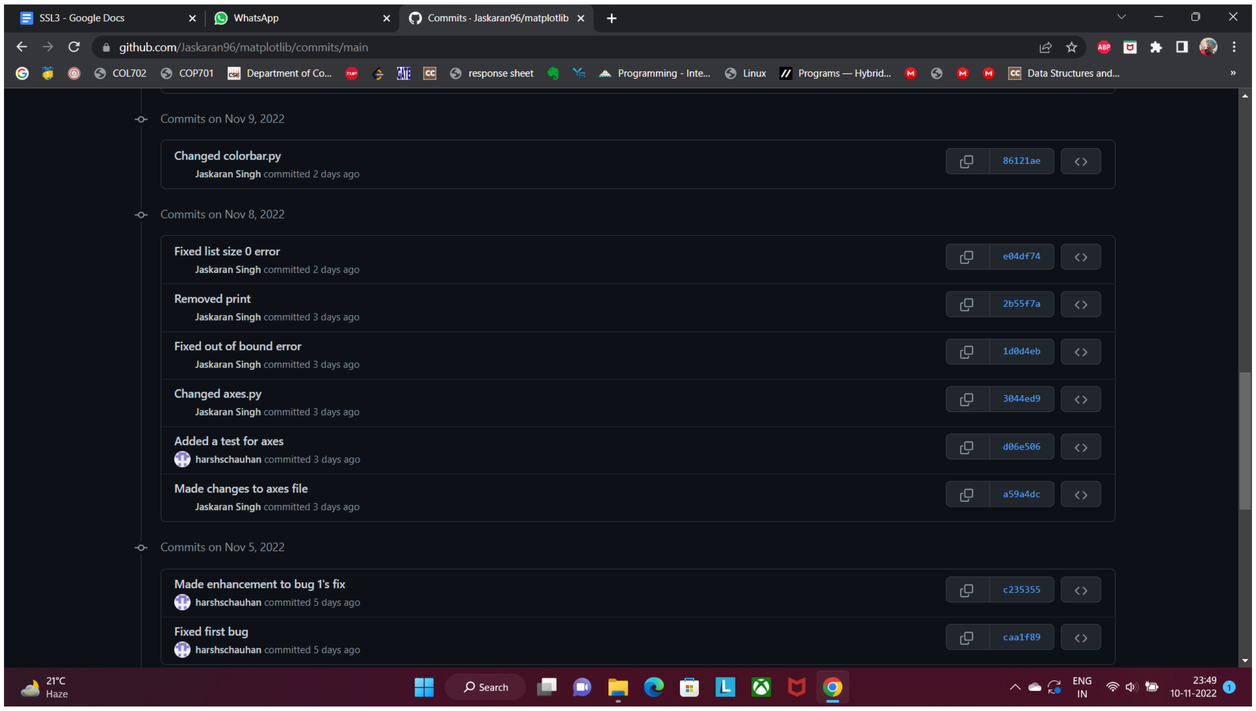

main

Commits on Nov 10, 2022

fixed contours battaladivyateja committed 25 seconds ago	edaf3e1
Improved scatter plot battaladivyateja committed 2 minutes ago	f453265
Added Sphinx support harshschauhan committed 42 minutes ago	f7b3294
Added another test for scatter plot harshschauhan committed 1 hour ago	e1487b8
added test for scatter plot (bug 3) harshschauhan committed 1 hour ago	cde9fa5
Added more image tests for contour plot harshschauhan committed 2 hours ago	6ffa25f
Adding fixes Jaskaran Singh committed 2 hours ago	9e161cd
Added build automation Jaskaran Singh committed 2 hours ago	f72e4c5
Added a contour plot test harshschauhan committed 3 hours ago	ecdf858
Fix legend.py Jaskaran Singh committed 3 hours ago	8b7c558



Added more image tests for contour plot harshschauhan committed 2 hours ago	6ffa25f
Adding fixes Jaskaran Singh committed 2 hours ago	9e161cd
Added build automation Jaskaran Singh committed 2 hours ago	f72e4c5
Added a contour plot test harshschauhan committed 3 hours ago	ecdf858
Fix legend.py Jaskaran Singh committed 3 hours ago	8b7c558
Fixed application crashing on string labelcolors Jaskaran Singh committed 4 hours ago	af798d3
Removed unnecessary tests harshschauhan committed 4 hours ago	7272554
Modified Test for bug 1 harshschauhan committed 5 hours ago	cff432b
updated documentation harshschauhan committed 11 hours ago	97a66c3
Merge branch 'main' of https://github.com/Jaskaran96/matplotlib harshschauhan committed 11 hours ago	9e8d974
fixed zorder issue harshschauhan committed 11 hours ago	1fba925
Adding sphinx docs Jaskaran Singh committed 12 hours ago	d4902ec



SSL3 - Google Docs | WhatsApp | Commits · Jaskaran96/matplotlib

github.com/Jaskaran96/matplotlib/commits/main

COL702 | COP701 | Department of Co... | response sheet | Programming - Inte... | Linux | Programs — Hybrid... | Data Structures and...

Commits on Nov 9, 2022

Changed colorbar.py
Jaskaran Singh committed 2 days ago — 86121ae

Commits on Nov 8, 2022

Fixed list size 0 error
Jaskaran Singh committed 2 days ago — e04df74

Removed print
Jaskaran Singh committed 3 days ago — 2b55f7a

Fixed out of bound error
Jaskaran Singh committed 3 days ago — 1d0d4eb

Changed axes.py
Jaskaran Singh committed 3 days ago — 3044ed9

Added a test for axes
harshschauhan committed 3 days ago — d06e506

Made changes to axes file
Jaskaran Singh committed 3 days ago — a59a4dc

Commits on Nov 5, 2022

Made enhancement to bug 1's fix
harshschauhan committed 5 days ago — c235355

Fixed first bug
harshschauhan committed 5 days ago — caa1f89

21°C Haze | Search | ENG IN | 23:49 10-11-2022

3.2) Auto Documentation using Sphinx

Sphinx makes it easy to create intelligent and beautiful documentation.

Here are some of Sphinx's major features:

- **Output formats:** HTML (including Windows HTML Help), LaTeX (for printable PDF versions), ePub, Texinfo, manual pages, plain text
- **Extensive cross-references:** semantic markup and automatic links for functions, classes, citations, glossary terms and similar pieces of information
- **Hierarchical structure:** easy definition of a document tree, with automatic links to siblings, parents and children
- **Automatic indices:** general index as well as a language-specific module indices
- **Code handling:** automatic highlighting using the Pygmets highlighter
- **Extensions:** automatic testing of code snippets, inclusion of docstrings from Python modules (API docs) via built-in-extensions and much more functionality via third-party extensions.
- **Themes:** modify the look and feel of outputs via creating themes, and re-use many third-party themes.
- **Contributed extensions:** dozens of extensions contributed by users; most of them installable from PyPI.

Sphinx uses the reStructuredText markup language by default, and can read MyST markdown via third-party extensions. Both of these are powerful and straightforward to use, and have functionality for complex documentation and publishing workflows. They both build upon Docutils to parse and write documents.

(Source - https://www.sphinx-doc.org/en/master)
For our project, we have documented the classes. Using Sphinx we are able to automatically generate HTML output format of our documentation.

3.3) Build Automation in Python

We have used **setuptools** to convert our application into a redistributable package.

All the configurations have been made in the setup.py file inside the main project folder.

Using the command :

```
python3 setup.py sdist bdist_wheel
```

we are able to create redistributable packages which are present under the “dist” sub-folder.

Now using **twine** we push the generated .whl and .tar.gz file into the the PyPI repository by using the following command :

```
twine upload dist/*
```

This uploads all the files generated inside the dist folder to the official PyPI reposity.

Now any one on the internet can install the package using pip install “package-name” and start the game.

Package name: matplotlibssl3

3.4) Unit Testing

Matplotlib uses the pytest framework.

The tests are in lib/matplotlib/tests, and customizations to the pytest testing infrastructure are in **matplotlib.testing**.

(source - https://www.guru99.com/pytest-tutorial.html#:~:text=PyTest%20is%20a%20testing%20framework,tests%20to%20complex%20functional%20tests.)

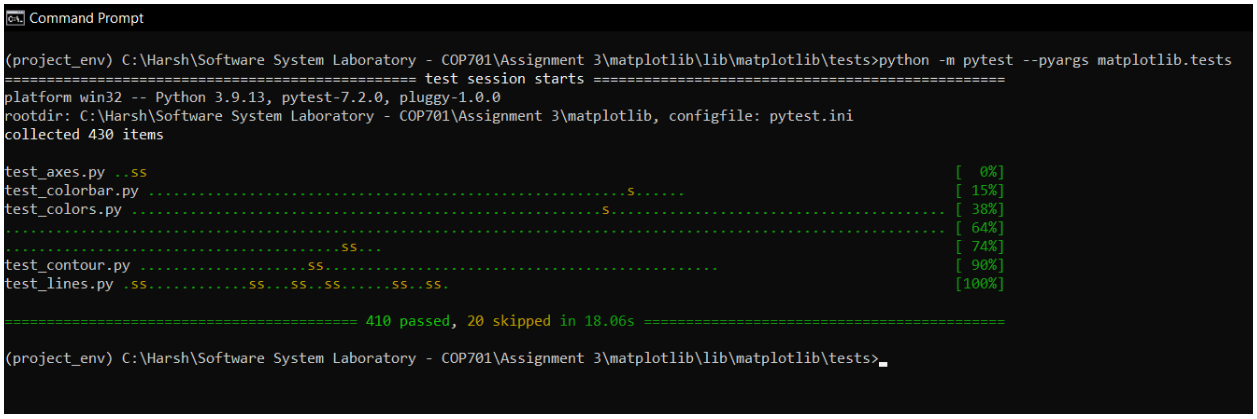
What is PyTest?

PyTest is a testing framework that allows users to write test codes using Python programming language. It helps you to write simple and scalable test cases for databases, APIs, or UI. PyTest is mainly used for writing tests for APIs. It helps to write tests from simple unit tests to complex functional tests.

Why use PyTest?

Some of the advantages of pytest are

- Very easy to start with because of its simple and easy syntax.
- Can run tests in parallel.
- Can run a specific test or a subset of tests
- Automatically detect tests
- Skip tests
- Open source



```
Command Prompt

(project_env) C:\Harsh\Software System Laboratory - COP701\Assignment 3\matplotlib\lib\matplotlib\tests>python -m pytest --pyargs matplotlib.tests
================================================= test session starts =================================================
platform win32 -- Python 3.9.13, pytest-7.2.0, pluggy-1.0.0
rootdir: C:\Harsh\Software System Laboratory - COP701\Assignment 3\matplotlib, configfile: pytest.ini
collected 430 items

test_axes.py ..ss                                                                                                 [  0%]
test_colorbar.py .........................................................s.....                                   [ 15%]
test_colors.py .......................................................s........................................ [ 38%]
................................................................................................................. [ 64%]
........................................ss...                                                                     [ 74%]
test_contour.py ....................ss...............................................                             [ 90%]
test_lines.py .ss...........ss...ss..ss......ss..ss.                                                              [100%]

=========================================== 410 passed, 20 skipped in 18.06s ===========================================

(project_env) C:\Harsh\Software System Laboratory - COP701\Assignment 3\matplotlib\lib\matplotlib\tests>
```

We added a few image tests in the test files already present in the matplotlib library in order to test the fix for the bugs mentioned above. We used image comparison and found if the test image and reference image were the same or not. Also, baseline testing for images was also used. Since there were many other tests for many other files and functions, we didn't run them completely since it was taking too much time and was unnecessary for our work.

4) <u>Contribution</u>

Jaskaran Singh

- Fixed Bug 1
- Fixed Bug 3
- GIT Version Control
- Build Automation
- Sphinx Documentation
- Uploading the distributable package to PyPI
- Project Report

Harsh Singh Chauhan

- Fixed Bug 2
- GIT Version Control
- SPHINX documentation
- Unit Testing
- Helped with Bug 3
- Project Report

Teja

- Helped with fixing bug 3
- GIT Version Control
- Automated SPHINX documentation
- Build Automation
- Unit Testing
- Project Report

5) **Bibliography**

- https://matplotlib.org/stable/devel/testing.html
- https://matplotlib.org/stable/devel/development_setup.html
- https://www.youtube.com/watch?v=APOPm01BVrk
- https://matplotlib.org/3.1.1/gallery/misc/zorder_demo.html
- https://www.pythonpool.com/matplotlib-zorder/
- https://www.sphinx-doc.org/en/master/
- https://pythonhosted.org/an_example_pypi_project/setuptools.html
- https://www.guru99.com/pytest-tutorial.html#:~:text=PyTest%20is%20a%20testing%20framework,tests%20to%20complex%20functional%20tests.