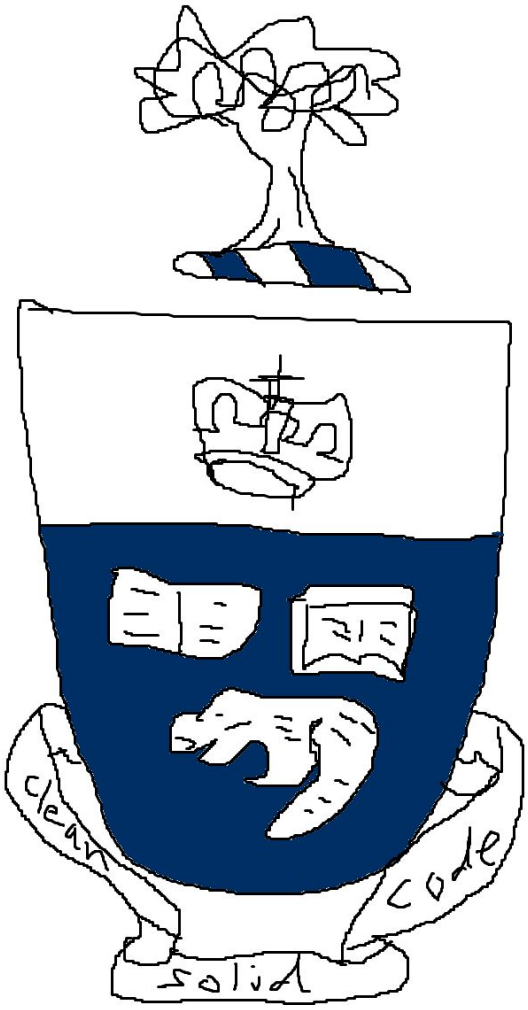


Deliverable 2p1: Team Plagiarism

Deliverable #2p1: Raising Issues
March 4th 2019
Team Members: Minqi Wang, Shuang Wu,
James Nicol, Xinrui Tong, Zixing Gong

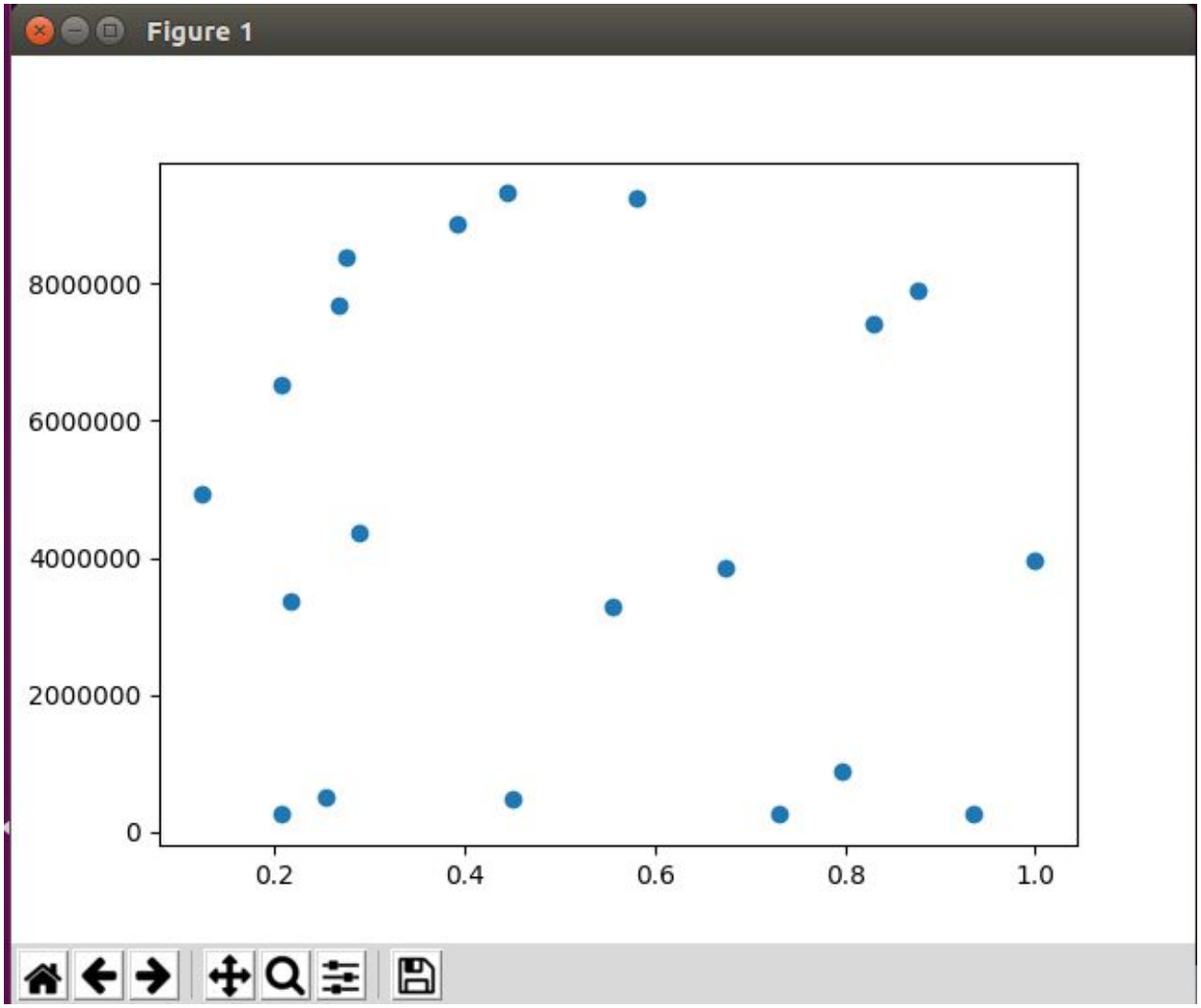
Table of Contents

Issue 1 - Jamie

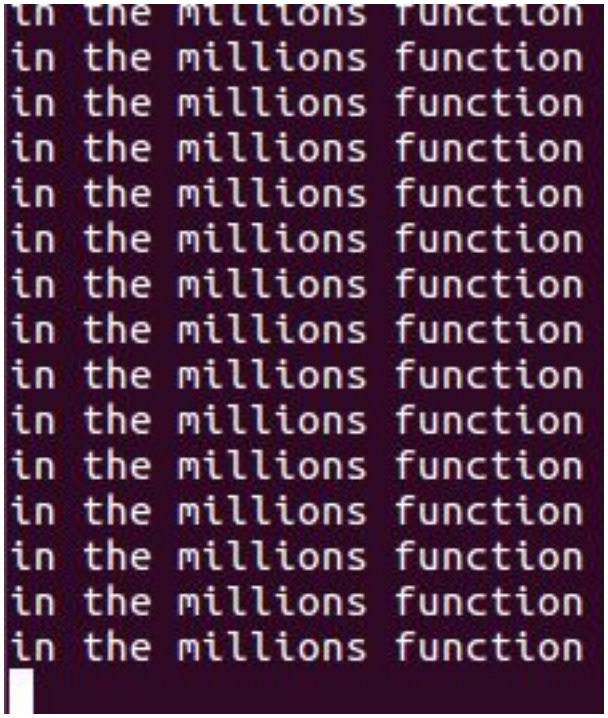
Issue Name: Add ‘fmt_r’ and ‘fmt_theta’ methods to polar axes
Issue #: 4568
URL: https://github.com/matplotlib/matplotlib/issues/4568
Type of Issue: Feature Request / API Consistency
Short Explanation:
There is no ax.fmt_ydata or ax.fmt_xdata functions when working on a polar graph compared to a regular graph. This issue is shown by using a function millions that returns the current y coordinates of the mouse pointer in millions. When calling it with a polar graph, millions is not called at all, since polar graphs use r and theta instead of x and y.
Idea of Solution:
There are two schools of thought for this issue: 1. add a fmt_rdata and fmt_thetadata functions for polar graphs, or 2. add fmt_xdata and fmt_ydata functions for polar graphs and map r and theta properly to x and y. (1) is suggested as being more accurate for the polar case, but risks making the implementation more confusing, while (2) seems to be preferred for consistencies sake.
Code Snippet:

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3
4  def millions(x):
5      print('in the millions function')              #confirmation that we are in the millions function
6      return '$%1.1fM' % (x*1e-6)
7
8  x = np.random.rand(20)
9  y = 1e7*np.random.rand(20)
10
11 fig, ax = plt.subplots()                           #millions does get called on regular graph
12 #fig, ax = plt.subplots(subplot_kw={'polar':True})  #millions does not get called on the polar graph
13 ax.fmt_ydata = millions
14 plt.plot(x, y, 'o')
15
16 plt.show()
```

This standard graph gives this as output:



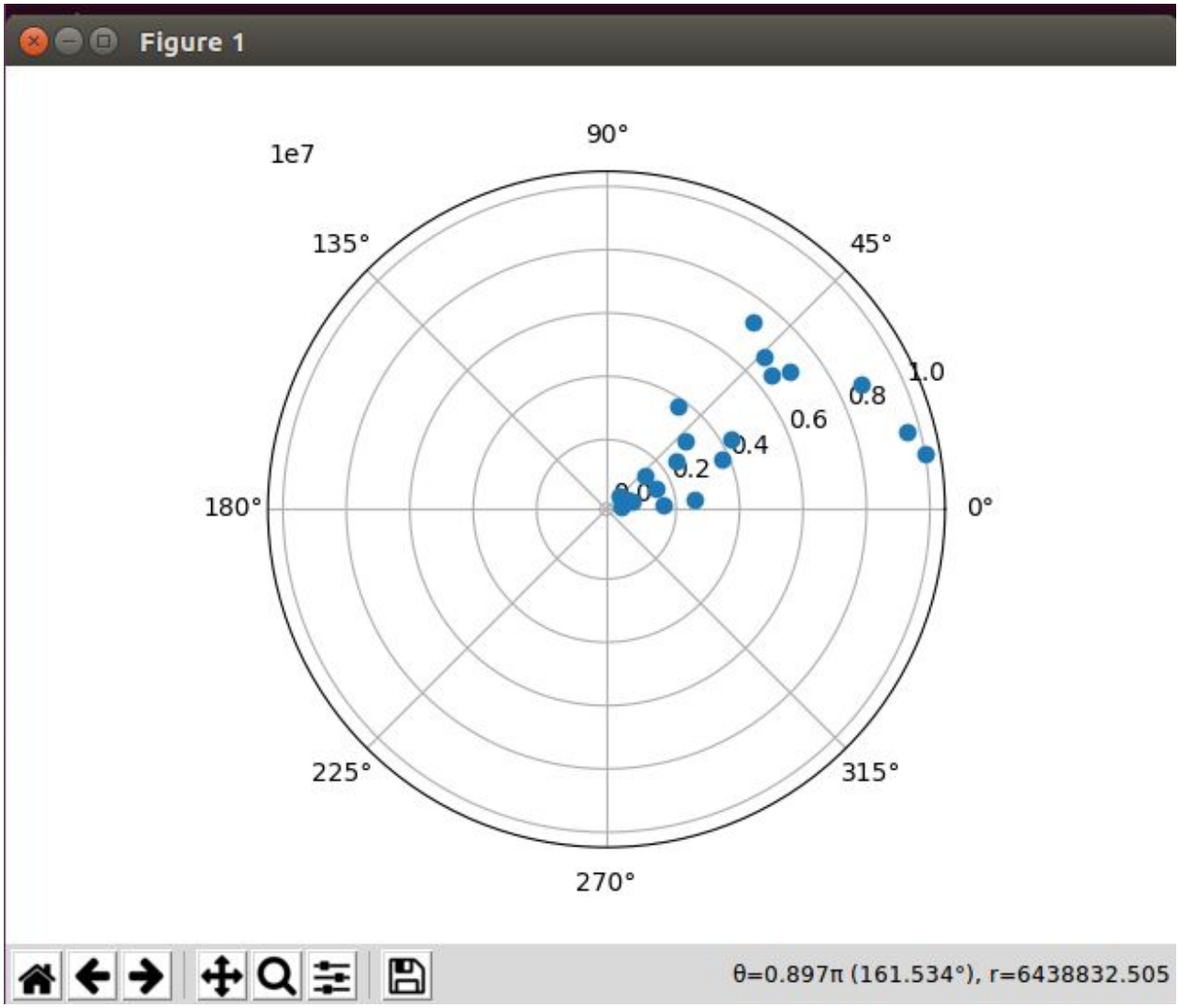
and mousing over the graph prints this:



Meanwhile, changing the subplot to this:

```
#fig, ax = plt.subplots()                          #millions does get called on regular graph
fig, ax = plt.subplots(subplot_kw={'polar':True})  #millions does not get called on the polar graph
```

Will give this as output:



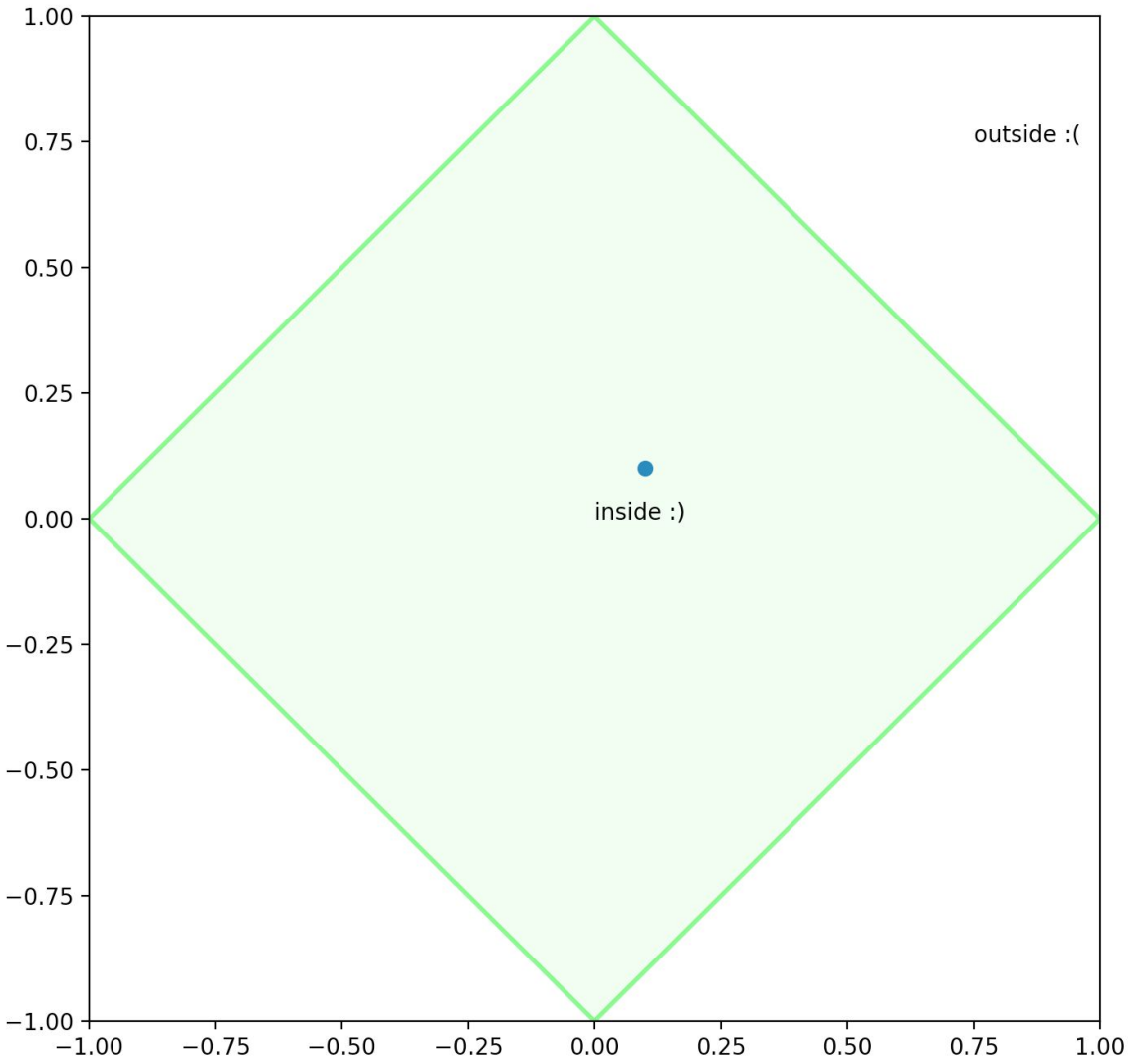
and mousing over the graph does not print anything in the terminal, meaning that millions is never called.

Issue 2 - Ken

Issue Name: text is not clipped by clip_path
Issue #: 8270
URL: https://github.com/matplotlib/matplotlib/issues/8270
Type of Issue: text
Short Explanation:
The matplotlib.text.Text class should have the capability of clipping the text to some specified path by taking the named arguments "clip_on=True/False" and "clip_path=somepath" in its constructor. The issue is that in some cases when "clip_on=True" and "clip_path=somepath" are set, the text outside the path still present. In the example below specifically, when adding 2 matplotlib.text.Text objects to ax (Axes), the firstly added one's clip path does not work.
Idea of Solution:
It seems that the possible cause of this problem is that the matplotlib.text.Text object's clip path gets dropped or reset at some point, so the solution of this problem is probably going through the related functions to find where the text's clip path gets dropped and then come up with a proper way to fix it. Starting with looking at the "_AxesBase.add_artist" and "Artist.set_clip_path" methods is suggested to resolve this bug.
Code Snippet:

```
import numpy as np
import matplotlib as mpl
from matplotlib import pyplot as plt

plt.figure(figsize=(8,8))
ax = plt.gca()

poly = mpl.patches.Polygon([[1,0], [0,1], [-1,0], [0,-1]],
                           facecolor="#ddffdd", edgecolor="#00ff00", linewidth=2, alpha=0.5)

ax.add_patch(poly)

# should not be displayed ...
txt_outside = mpl.text.Text(0.75,0.75,"outside :(", clip_on=True, clip_path=poly)
ax.add_artist(txt_outside)

# should be (and is) displayed
txt_inside = mpl.text.Text(0.,0.,"inside :)", clip_on=True, clip_path=poly)
ax.add_artist(txt_inside)

# works perfectly
scatter = plt.scatter(*np.transpose([[0.1,0.1],[0.7,0.7]]), zorder=5, clip_path=poly)

ax.set_xlim(-1,1)
ax.set_ylim(-1,1)
plt.show()
```

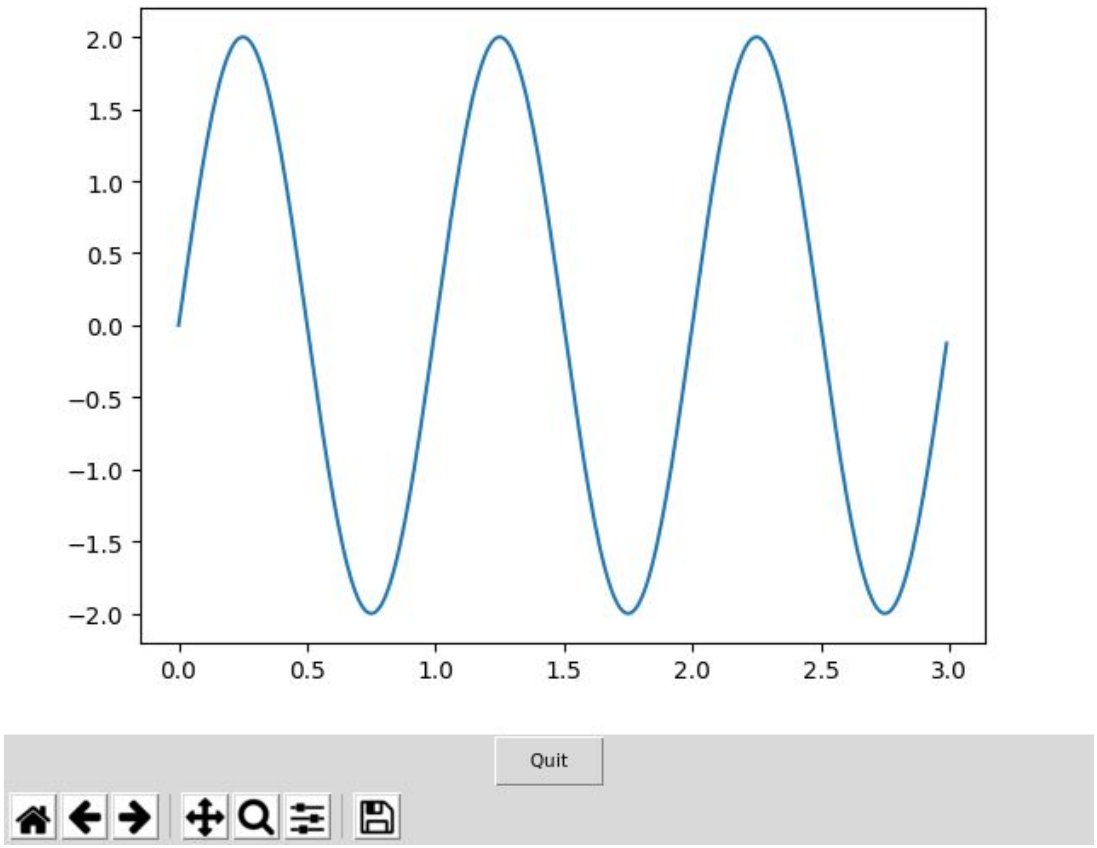
In the above code we set all of the text “inside :)”, the text “outside :(” and the scatter plot to be clipped to the quadrilateral. The code gives the following output:



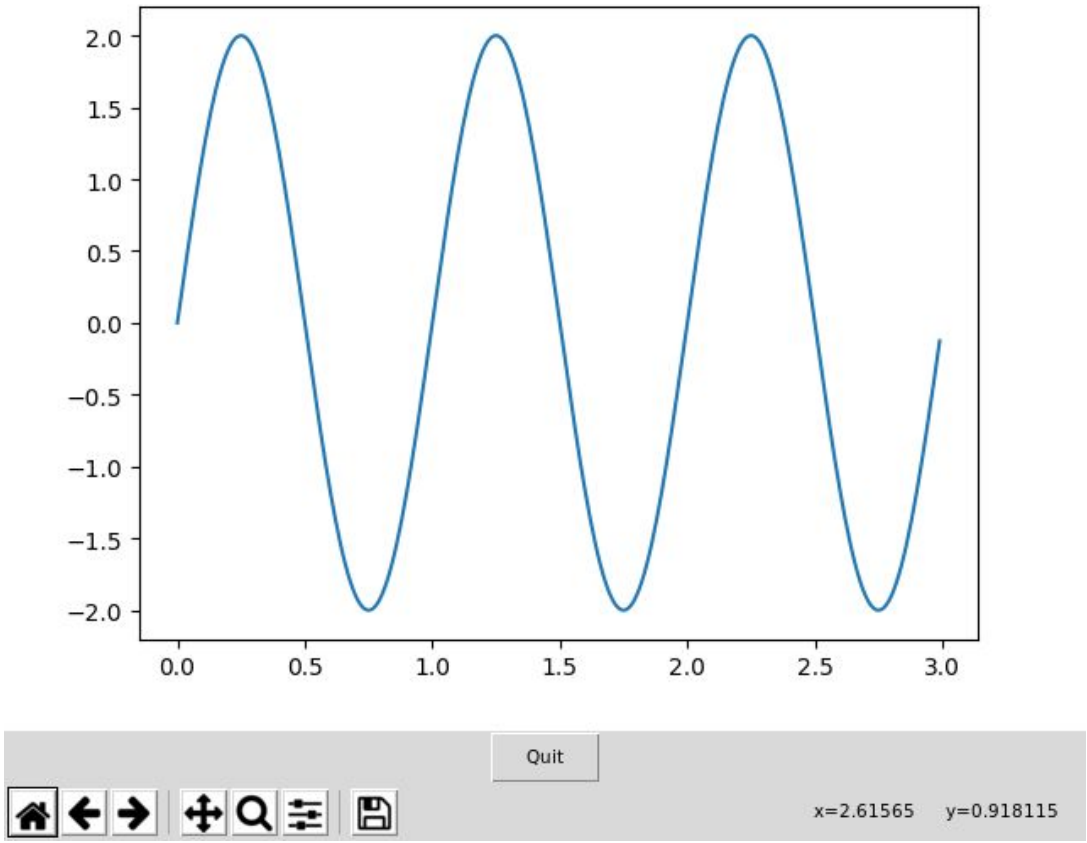
As we can see the text “inside :)” is presented as expected, because it sits inside the quadrilateral. However, the text “outside :(” is also presented on the graph, which is not desirable because it sits outside the quadrilateral and we have set it to be clipped to the quadrilateral. The scatter works well as expected.

Issue 3 - Zixing

Issue Name: Matplotlib keymap stop working after pressing tab
Issue #: 13484
URL: https://github.com/matplotlib/matplotlib/issues/13484
Type of Issue: Feature Request
Short Explanation:
When using the example on embedding matplotlib in tk in the link below (Code Snippet section), if the user presses the tab key, the focus is shifted to the next widget, disabling keyboard interactivity with the mpl plot until tab is pressed enough times to shift the focus back onto the plot. Note that clicking on the plot will not restore the keyboard interactivity, which is what the feature request is for: to change mpl to set the focus to the clicked widget when using mpl with tkinter (and other applicable 3rd party GUIs)
Idea of Solution:
mpl should be able to get the selected widget using the matplotlib.backend_bases.Event.guiEvent attribute, then we can configure the FigureCanvasTkAgg class to set focus to the selected widget.
Code Snippet:
https://matplotlib.org/gallery/user_interfaces/embedding_in_tk_sgskip.html



Focus is on the plot



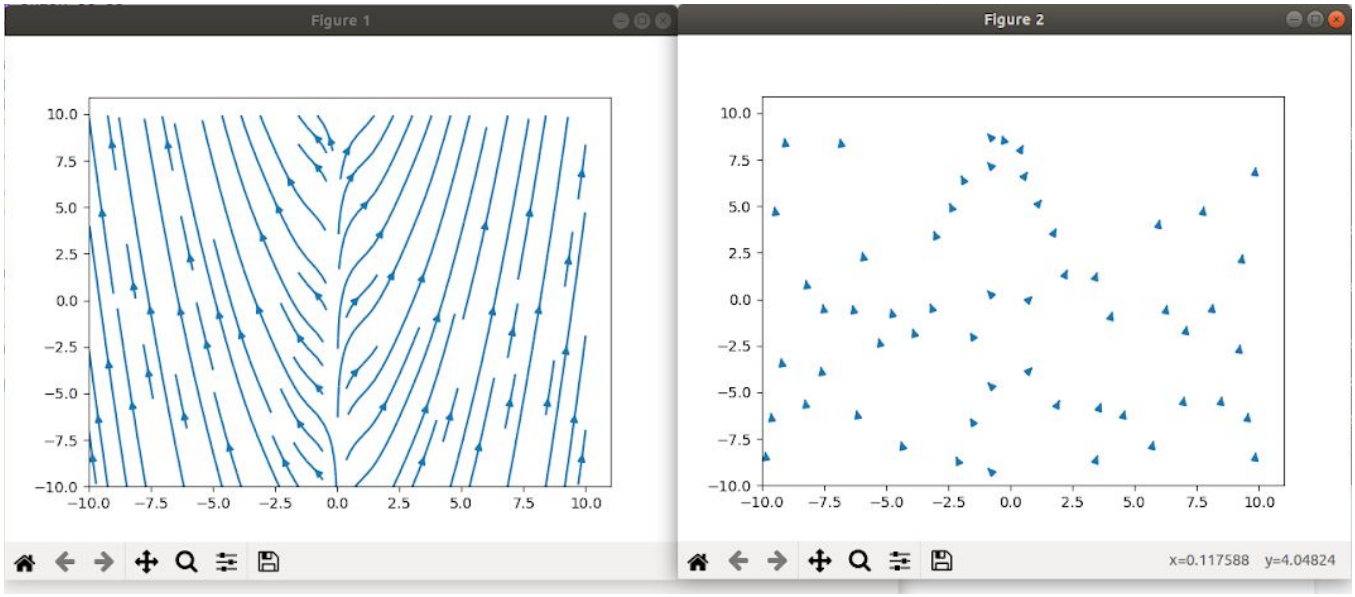
Focus is on the home button

Issue 4 - Xinrui

Issue Name: Matplotlib keymap stop working after pressing tab
Issue #: 6000
URL: https://github.com/matplotlib/matplotlib/issues/6000
Type of Issue: Confirmed Bug
Short Explanation:
The 'set_visible' method is not working as intended for PatchCollection as a result of objects being added as individuals to a collection and then when returned it returns the collection instead of individuals. Thus changes to visibility does not render in the final result as expected
Idea of Solution:
A possible solution is to return individuals for arrows instead of a collection or break down the collection into individual to ensure 'set_visible' works as intended.
Code Snippet:

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

# Data
x = np.linspace(-10, 10, 10)
y = np.linspace(-10, 10, 10)
X, Y = np.meshgrid(x, y)
U = X
V = X**2

# basic streamline plot
plt.figure()
sp1 = plt.streamplot(x, y, U, V)

# Attempt to hide lines and arrows from streamline plot
plt.figure()
sp2 = plt.streamplot(x, y, U, V)
sp2.lines.set_visible(False)
sp2.arrows.set_visible(False)

plt.show()
```

The above code snippet produces the following:

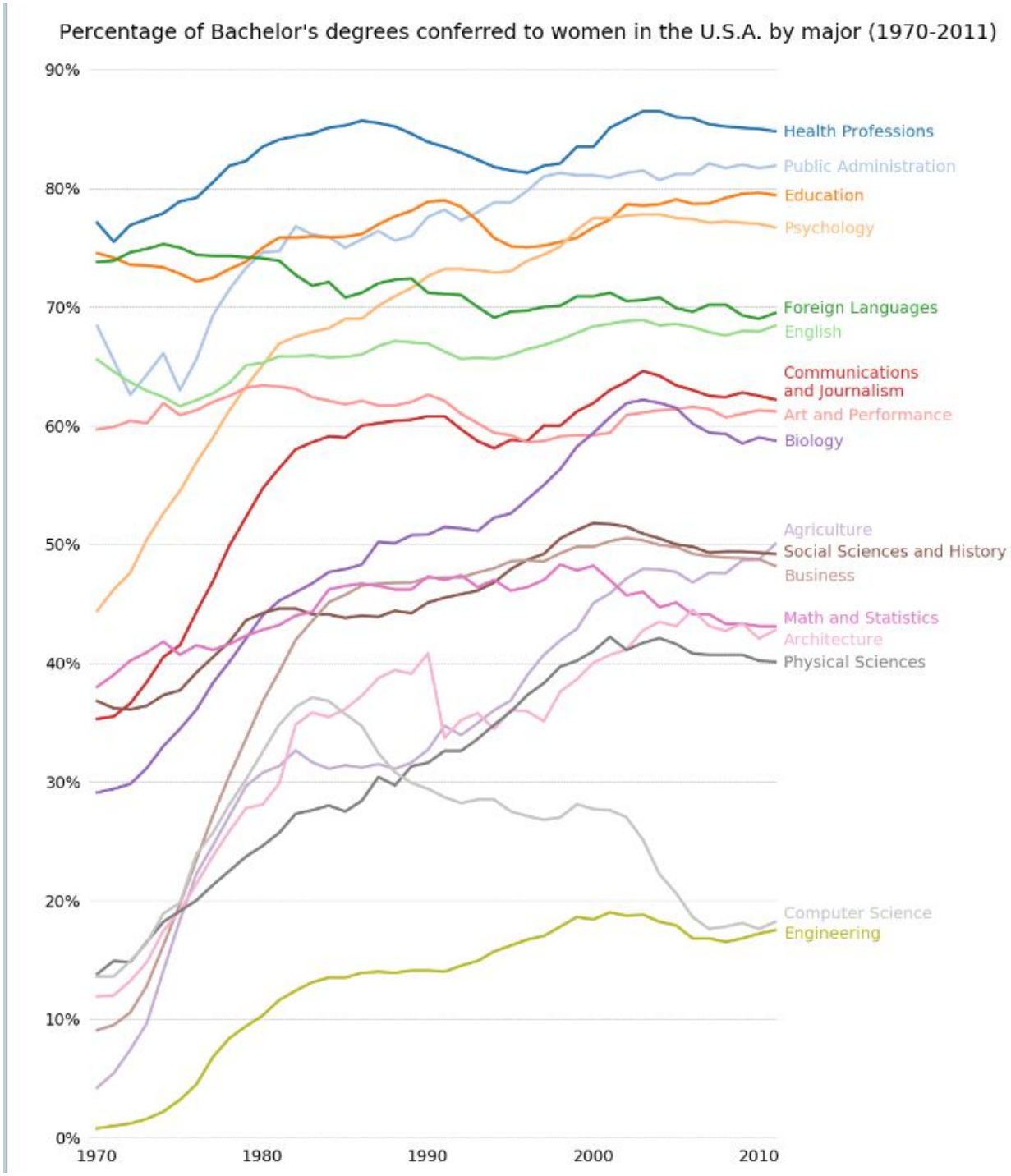


Notice that Figure 2 still has arrows when it should be blank.

Issue 5 - Shuang

Issue Name: Option to place legend labels near to the data

Issue #: 12939
URL: https://github.com/matplotlib/matplotlib/issues/12939
Type of Issue: Feature Requested / Wishlist
Short Explanation:
The current way to add legend to label near to the line is complicated, OP would like to have a simpler API to accomplish this.
Idea of Solution:
This is actually not a bug but a feature requested from the OP. According to the comments below this opening issue, contributor jklymak had suggested implementing a new method instead of using .legend() API. Member of Matplotlib, QuLogic, provided a link to an example as a potential basis for the newly method: https://matplotlib.org/gallery/showcase/bachelors_degrees_by_gender.html. Therefore, we may use this as our starting point to implementing this feature.
Feature Concept:
Giving a new API, called label_line(), automatically put all the text labels to the end of the corresponding line, with same color to the line.
What would makes the label near to the line looks like:



Notice the labels are to the end of the line, with same color to their corresponding line and ordered by the last Y-axis data from high to low.