How to Easily Reposition Your Legend in Matplotlib

Authored by

stats writer

December 3, 2025

Mastering data visualization requires not only generating accurate plots but also ensuring all elements, such as the Matplotlib legend, are positioned optimally for clarity and aesthetic appeal. The legend acts as a critical guide, mapping visual cues (like colors or line styles) to the datasets they represent.

While Matplotlib is often effective at automatically placing the legend, there are many scenarios where manual adjustment is necessary to prevent obstruction of key data points or to place the legend outside the plot area entirely. This guide details the essential techniques for repositioning the legend using the primary function: **plt.legend()**.

There are two primary methods for controlling the placement of the legend in Matplotlib visualizations: utilizing predefined location strings via the `loc` parameter, or employing precise coordinate control using the `bbox_to_anchor` argument for custom or external placement.

To precisely control the positioning of the legend in Matplotlib visualizations, we primarily interact with the **plt.legend()** function.

Controlling Legend Position with the `loc` Parameter

The most straightforward approach to setting the legend position is using the `loc` parameter within the plt.legend() function. The `loc` parameter accepts specific strings or integer codes that correspond to one of the nine standard areas within the axes bounding box, plus a special tenth option for automatic placement.

For instance, if you wish to fix the legend to the top left area of your plotting region, the corresponding string argument is utilized as shown in the following syntax snippet:

plt.legend(loc='upper left')

It is important to recognize the default behavior of the `loc` argument. By default, `loc` is set to `'best'`. When `loc='best'` is specified, Matplotlib algorithmically determines the optimal placement for the legend by iterating through all possible locations and choosing the one that minimally overlaps with any displayed data points, ensuring maximum data visibility.

Predefined Positional Keywords for Legend Placement

When customizing the legend placement using the `loc` parameter, you have access to a standardized set of positional keywords. These keywords correspond directly to specific regions within the plot's axes bounding box. Using these strings provides a quick and robust way to ensure the legend appears in a predictable location.

The following list details all the standard string arguments available for the `loc` parameter, offering comprehensive control over internal legend positioning:

upper right: Places the legend anchor point in the top right corner.
upper left: Places the legend anchor point in the top left corner.
lower left: Places the legend anchor point in the bottom left corner.
lower right: Places the legend anchor point in the bottom right corner.
right: Aligns the legend along the right vertical edge.
center left: Centers the legend vertically along the left edge.
center right: Centers the legend vertically along the right edge.
lower center: Centers the legend horizontally along the bottom edge.
upper center: Centers the legend horizontally along the top edge.
center: Centers the entire legend within the plot area.

It is beneficial to test several of these predefined locations to determine which one best complements the specific data distribution of your graph, especially before resorting to manual coordinate specification.

Precise Control using the `bbox_to_anchor` Argument

For scenarios demanding precise placement, particularly when positioning the legend outside the standard plotting area, the `bbox_to_anchor` argument becomes essential. This argument accepts a tuple of coordinates (x, y) that define a specific point on the figure where a designated corner of the legend should be anchored. This provides ultimate flexibility beyond the ten predefined locations.

When using `bbox_to_anchor`, the coordinates are generally normalized relative to the axes, where (0, 0) is the lower-left corner of the axes and (1, 1) is the upper-right corner. To place the legend outside the plot boundary, coordinates exceeding 1.0 (or less than 0.0) are necessary.

Crucially, `bbox_to_anchor` must be used in conjunction with the `loc` parameter. The `loc` value determines which part of the legend box (e.g., its upper left corner, its center, etc.) is placed **at** the coordinates specified by `bbox_to_anchor`. For instance, to place the legend just outside the top right of the plot, we might use coordinates slightly greater than (1, 1) and instruct the `'upper left'` corner of the legend box to snap to that coordinate:

plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0)

In this example, (1.05, 1) specifies a location slightly to the right of the plot's top boundary. By setting `loc='upper left'`, we instruct Matplotlib to position the upper-left corner of the legend

box at this external coordinate. The `borderaxespad=0` ensures minimal padding between the axis boundary and the anchor point. These techniques are demonstrated practically in the following dedicated examples.

Example 1: Positioning the Legend Inside the Plot Boundary

We begin with practical examples demonstrating the use of the standard `loc` parameter to control internal placement. This is the most common use case for quick adjustments when the default `'best'` location is unsuitable. For this example, we utilize pandas to structure the sample data and generate a basic line plot using Matplotlib's `pyplot` interface.

The code below illustrates how to generate a plot comparing 'points' and 'assists' data, explicitly forcing the legend to reside in the `'center right'` section of the plot area. Note the addition of the `title` argument for enhanced clarity.

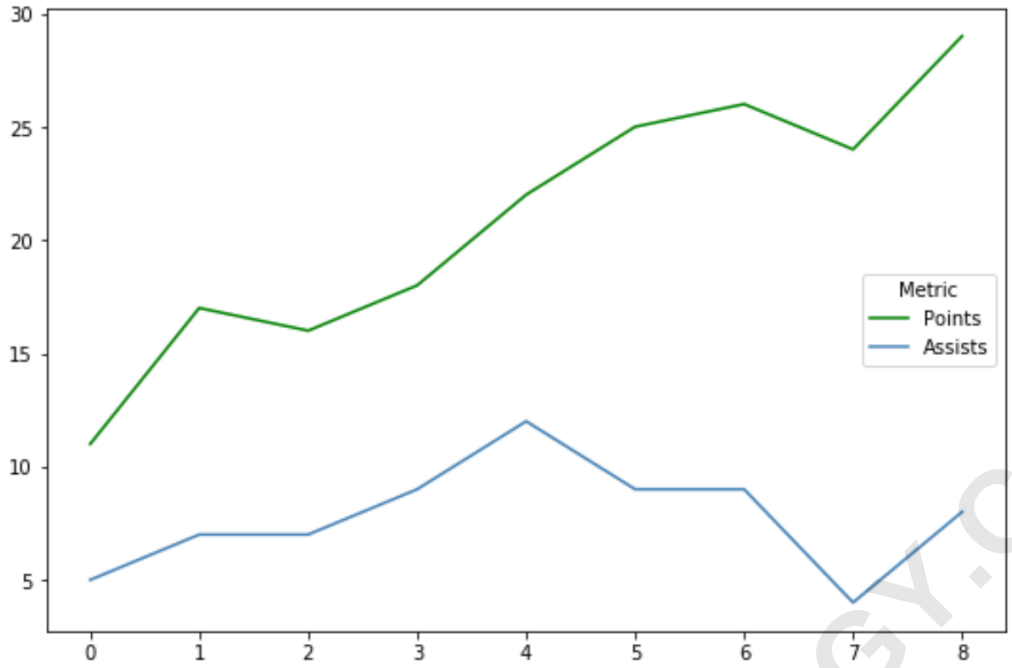
```
import pandas as pd
import matplotlib.pyplot as plt

#create data
df = pd.DataFrame({'points': ,
'assists': })

#add lines to plot
plt.plot(df, label='Points', color='green')
plt.plot(df, label='Assists', color='steelblue')

#place legend in center right of plot
plt.legend(loc='center right', title='Metric')
```

The resulting visualization confirms that the legend is successfully constrained to the center-right boundary of the plot, demonstrating the immediate effect of the `loc` parameter:



Internal Placement Variation: Utilizing the 'upper left' Position

To further illustrate the ease of using predefined location strings, this variation of the previous example demonstrates switching the legend's location to the opposite corner. This is crucial when the data density shifts, requiring relocation of static elements to maintain clarity.

By simply changing the `loc` parameter value from `'center right'` to `'upper left'`, we instruct Matplotlib to anchor the legend box accordingly. All other components of the plot generation remain identical, highlighting the modularity of the `loc` argument in **plt.legend()**.

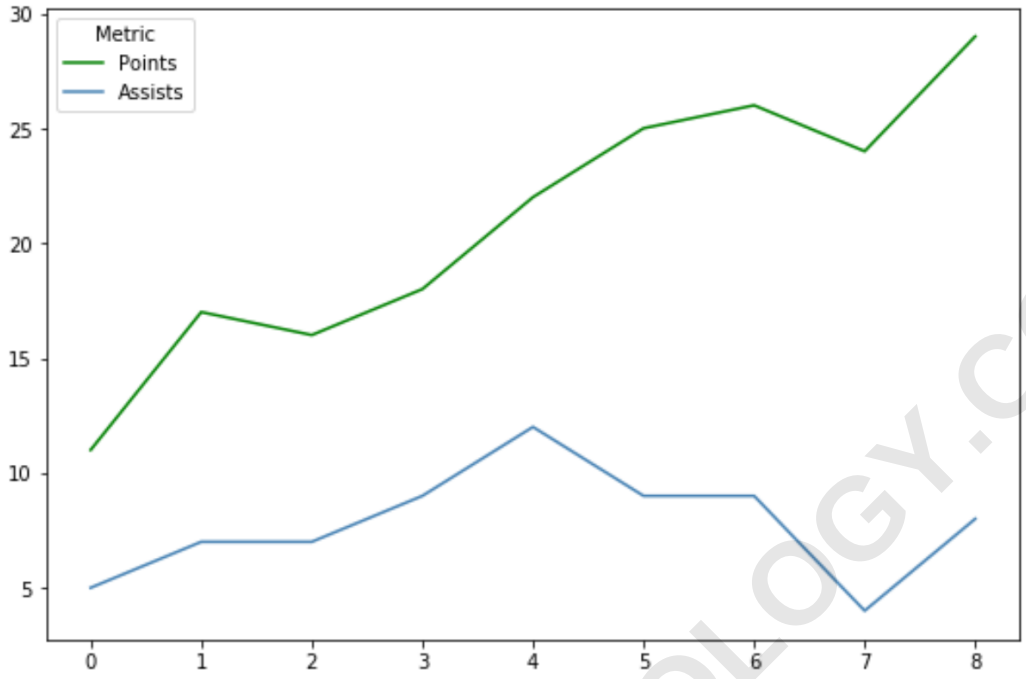
import pandas as pd
import matplotlib.pyplot as plt

#create data
df = pd.DataFrame({'points': ,
'assists': })

#add lines to plot
plt.plot(df, label='Points', color='green')
plt.plot(df, label='Assists', color='steelblue')

#place legend in center right of plot
plt.legend(loc='upper left', title='Metric')

As shown in the output image, the legend is now correctly positioned in the upper-left quadrant, demonstrating how simple string modifications achieve immediate positional shifts:



Example 2: Leveraging `bbox_to_anchor` for External Placement (Top Right)

When the legend obscures vital data points or if the design requires the plot to be self-contained, positioning the legend entirely outside the axes is the appropriate solution. This requires using the `bbox_to_anchor` parameter, which mandates coordinate inputs relative to the figure or axes.

To place the legend just outside the top-right corner, we specify coordinates slightly greater than (1, 1). In the code below, we use `(1.02, 1)`. Here, the X-coordinate of 1.02 ensures the legend starts just past the right edge (X=1), while the Y-coordinate of 1 aligns it perfectly with the top edge of the plot. We combine this with `loc='upper left'`, meaning the upper-left corner of the legend box is what snaps to the (1.02, 1) anchor point.

import pandas as pd
import matplotlib.pyplot as plt

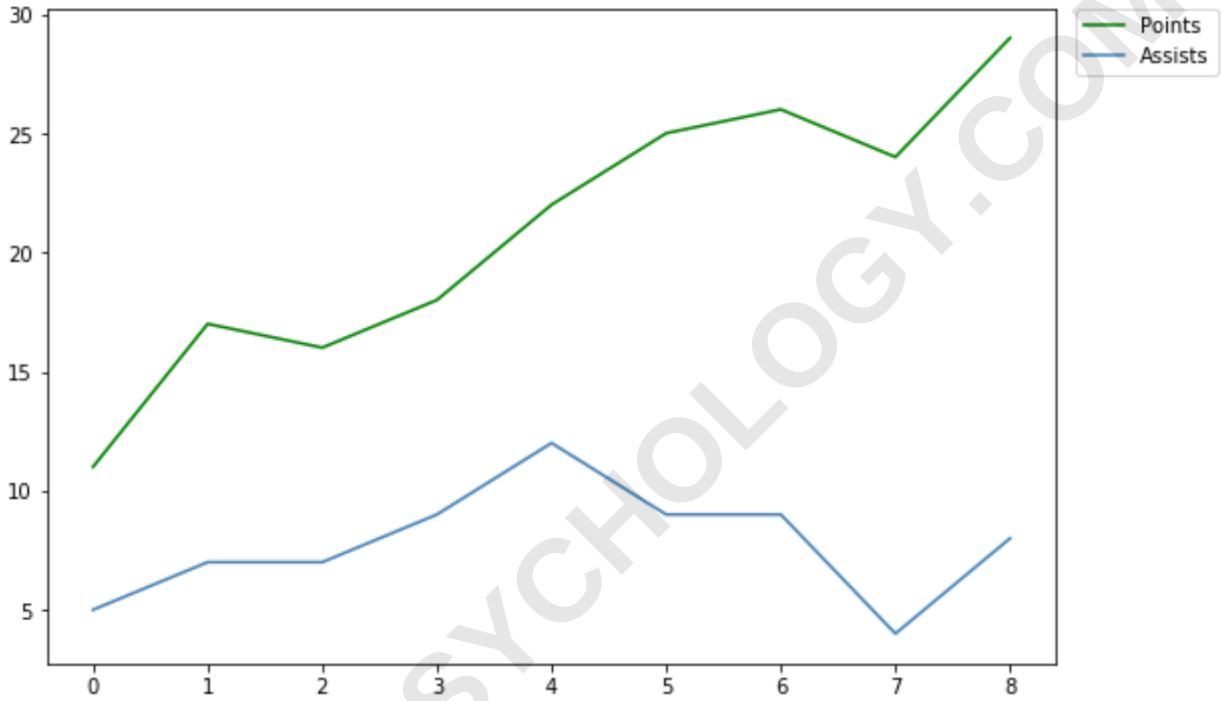
#create data
df = pd.DataFrame({'points': ,
'assists': })

#add lines to plot
plt.plot(df, label='Points', color='green')

```
plt.plot(df, label='Assists', color='steelblue')

#place legend outside top right corner
plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0)
```

The resulting visualization clearly shows the legend moved out of the plotting area, using the space provided to the right of the Y-axis:



Example 3: Controlling Vertical Alignment with `bbox_to_anchor` (Bottom Right)

Building upon the previous example, we can easily shift the external legend vertically by adjusting only the Y-coordinate within the `bbox_to_anchor` tuple. This demonstrates the fine-grained control available when using absolute coordinates.

To place the legend near the bottom right of the plot boundary, we maintain the X-coordinate at `1.02` (to keep it just outside the right edge) but change the Y-coordinate to a small value, such as `0.1`. The combination `bbox_to_anchor=(1.02, 0.1)`, coupled with `loc='upper left'`, ensures the legend's top-left corner aligns near the bottom right of the axes.

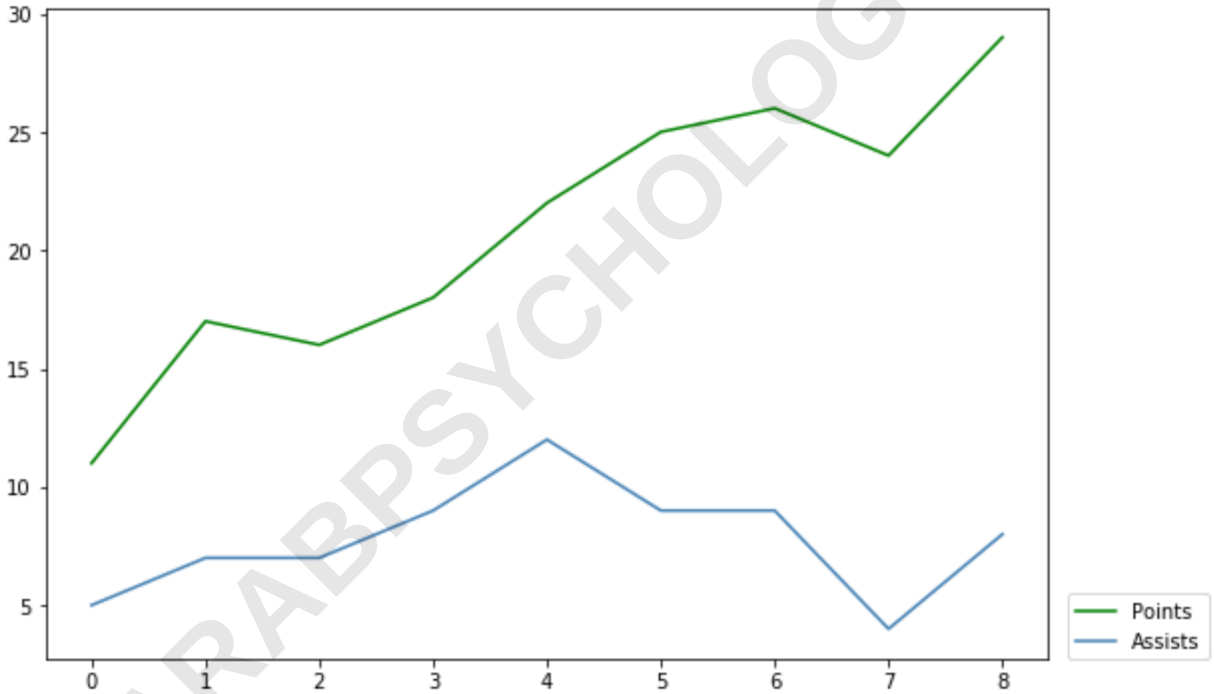
```
import pandas as pd
import matplotlib.pyplot as plt
```

```
#create data
df = pd.DataFrame({'points': ,
'assists': })

#add lines to plot
plt.plot(df, label='Points', color='green')
plt.plot(df, label='Assists', color='steelblue')

#place legend outside bottom right corner
plt.legend(bbox_to_anchor=(1.02, 0.1), loc='upper left', borderaxespad=0)
```

Observe how the minimal change in the Y-coordinate successfully shifts the legend to the bottom edge of the visualization space:



Understanding Coordinate Systems and `loc` Interaction

A frequent point of confusion when using `bbox_to_anchor` is understanding the interplay between the two key parameters. The `bbox_to_anchor` parameter specifies a single anchor point (x, y) in the normalized coordinate system (relative to the axes). However, this point does not define where the legend begins; rather, it defines where one of the legend's internal anchor points is placed.

The `loc` parameter dictates which corner or edge of the legend bounding box will be fixed to the anchor point defined by `bbox_to_anchor`. For example, if you set `bbox_to_anchor=(1, 1)` and

`loc='upper left'`, the legend's upper-left corner will be placed exactly at the plot's upper-right corner. Conversely, if you set `bbox_to_anchor=(1, 1)` and `loc='lower right'`, the legend's lower-right corner will be placed exactly at the plot's upper-right corner, causing the legend to appear mostly below and to the left of that point.

For developers seeking a more in-depth technical dive into the coordinate transformation mechanism within Matplotlib, we highly recommend referring to the official documentation for a detailed explanation of the bbox_to_anchor() argument and its coordinate handling. Mastering this relationship is key to achieving sophisticated and complex legend placements.

Summary of Key Legend Position Parameters

Successful legend positioning depends on selecting the right tools for the job. For simple adjustments within the plot, the `loc` string parameters are generally sufficient. For cases requiring the legend to be outside the plot area, or if extremely precise positioning is needed, `bbox_to_anchor` must be employed alongside `loc`.

The following list summarizes the primary parameters discussed for legend positioning in Matplotlib:

loc (Location): Accepts predefined strings (e.g., 'upper left', 'center') or integer codes (0-10). Determines which corner of the axes the legend should be placed in, or which corner of the legend box should align with the `bbox_to_anchor` point.
bbox_to_anchor: Accepts a tuple (x, y) defining the anchor point in normalized axes coordinates (0 to 1). If coordinates fall outside the range , the legend is placed externally. Requires the `loc` parameter to define the legend's alignment relative to this anchor point.
borderaxespad: Controls the padding between the axes edge and the legend when `bbox_to_anchor` is used. A value of 0 minimizes this padding.

By mastering these three parameters, any user of Python visualization libraries can ensure their plots are both informative and aesthetically polished, avoiding the common issue of critical data being obscured by the legend box. For additional customization options, such as adding descriptive labels, you may refer to resources detailing how to add a title to the legend in Matplotlib.

How to Add a Title to Legend in Matplotlib