Change the Position of a Legend in Matplotlib

Authored by

Mohammed looti

November 2, 2025

One of the most critical elements in creating effective data visualization is ensuring that all graphical components are clearly labeled and do not obscure the underlying data. In the Matplotlib library, managing the position of the plot Legend is essential for producing professional and readable figures.

To precisely control the location of a legend within a Matplotlib plot, developers utilize the **plt.legend()** function. This function offers robust parameters that allow the legend to be placed either inside the plotting area using predefined locations or entirely outside the plot using normalized coordinate systems.

Understanding the Matplotlib Legend Mechanism

The **plt.legend()** function is the primary tool for displaying descriptive labels associated with plotted data series. While Matplotlib attempts to automatically select an optimal position, manual control is frequently necessary to avoid overlapping lines or points, especially in complex visualizations.

The most straightforward way to adjust the legend's placement is by supplying the `loc` parameter, which accepts specific string arguments corresponding to standard positions on the plot canvas.

For instance, if you wished to ensure the legend appears in the upper left region of the graph, you would apply the following syntax to the **plt.legend()** call:

plt.legend(loc='upper left')

Understanding these positional parameters is key to mastering data presentation using this widely adopted Python library.

Default Legend Placement: The "Best" Location

By default, if no `loc` parameter is explicitly specified when calling **plt.legend()**, Matplotlib employs a smart positioning algorithm.

The default location setting is `"best"`. This instructs the plotting engine to intelligently search for an area within the axes where the legend box minimizes overlap with existing data elements (lines, markers, or bars). This automatic detection often works well for simple plots, but it can sometimes fail to find an ideal spot, or it may select a position that conflicts with the desired aesthetic layout.

When the `"best"` setting is insufficient, or when a specific visual standard must be maintained across multiple plots, manual specification of the location becomes necessary.

Specifying Internal Legend Locations Using the plt.legend() `loc` Parameter

When placing the legend inside the plot area, Matplotlib provides ten standard string locations. These strings offer quick access to common corner and edge positions, ensuring the legend is anchored to a specific part of the axes.

By defining the `loc` argument with one of these strings, you fix the anchor point of the legend box relative to the plot area. For example, selecting `'upper right'` places the legend box such that its top-right corner aligns with the top-right corner of the plot axes.

You can specify any of the following standard string locations for internal legend placement:

upper right
upper left
lower left
lower right
right
center left
center right
lower center
upper center
center

These options cover virtually every internal quadrant and edge position, providing ample flexibility for ensuring the legend is visible without manually calculating coordinates.

Practical Example 1: Adjusting Legend Position Inside the Plot

The following examples demonstrate how to create a simple line plot using Pandas data and then manipulate the internal location of the legend using different `loc` parameters.

First, we will initialize the data and plot two lines representing hypothetical "points" and "assists." We then place the legend inside the `'center right'` portion of the Matplotlib plot:

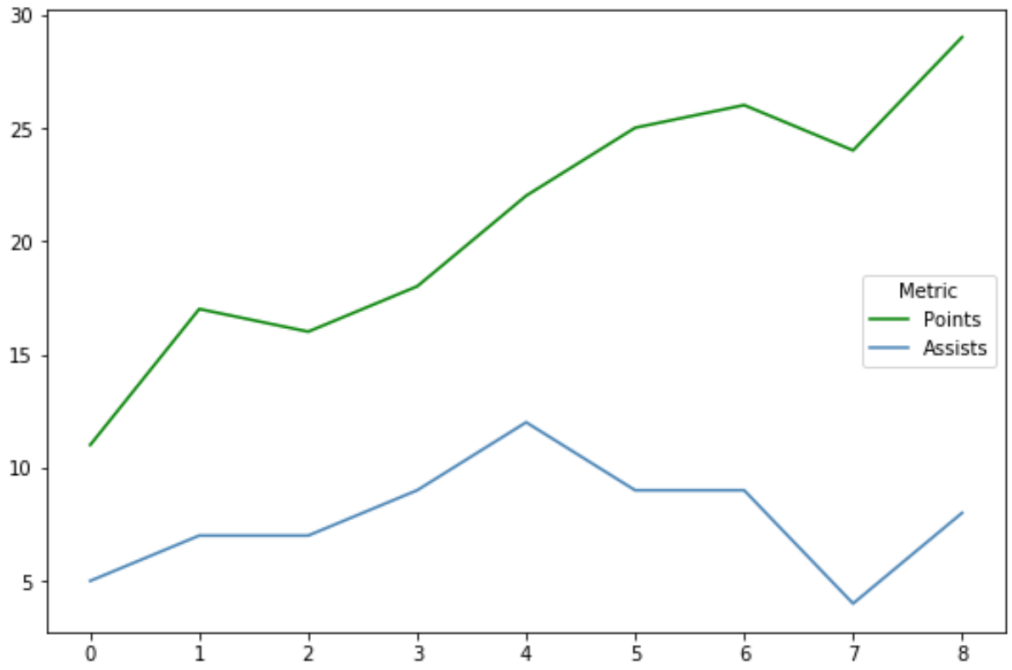
import pandas as pd
import matplotlib.pyplot as plt

#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')
```



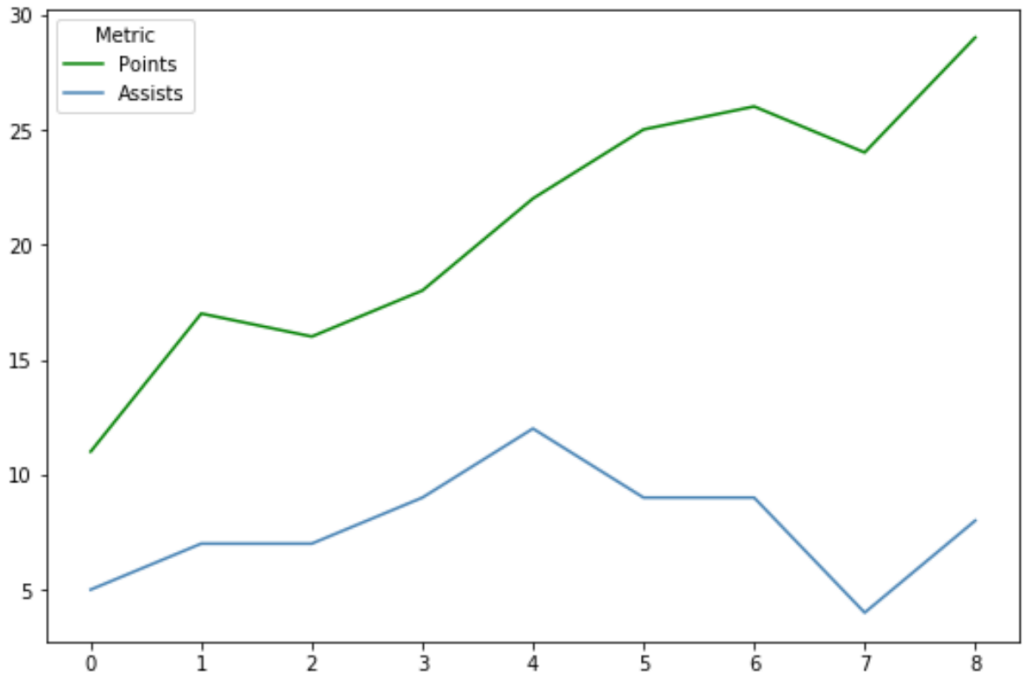
Alternatively, the following code snippet demonstrates how easily we can shift the exact same legend to the `'upper left'` corner of the visualization, simply by changing the value of the `loc` parameter:

```
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')
```



Mastering External Placement with bbox_to_anchor()

For layouts where the legend must reside completely outside the plotting area--for instance, to maximize data visibility or integrate the plot into a complex document structure--the `bbox_to_anchor()` argument is indispensable. This powerful tool allows users to define an arbitrary bounding box outside the traditional axes boundaries.

The `bbox_to_anchor()` parameter expects a tuple of coordinates, typically `(x, y)`, which define the anchor point for the legend box. Crucially, these coordinates are defined in normalized axes coordinates, meaning that (0, 0) is the lower-left corner of the plot axes, and (1, 1) is the upper-right corner.

To place the legend outside the plot, we typically use coordinates greater than 1.0 (for right or top placement) or less than 0.0 (for left or bottom placement).

When utilizing `bbox_to_anchor()`, it is vital to pair it with the `loc` parameter. In this context, `loc` defines which part of the legend box should be anchored to the coordinates specified by `bbox_to_anchor()`. For example, if `bbox_to_anchor=(1.05, 1)` is set, and `loc='upper left'` is used, the upper-left corner of the legend box will be placed at the point (1.05, 1) on the normalized axes, thus positioning the entire legend slightly outside the top-right corner.

Consider the following syntax for positioning the legend outside the top right corner of the plot:

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

The additional parameter, `borderaxespad=0`, controls the padding between the axes edge and the legend. Setting it to zero removes any automatic spacing, providing precise control over the placement.

Practical Example 2: Placing the Legend Outside the Plot

Using the same underlying data structure (a DataFrame containing points and assists), we can now demonstrate how `bbox_to_anchor()` facilitates external legend placement.

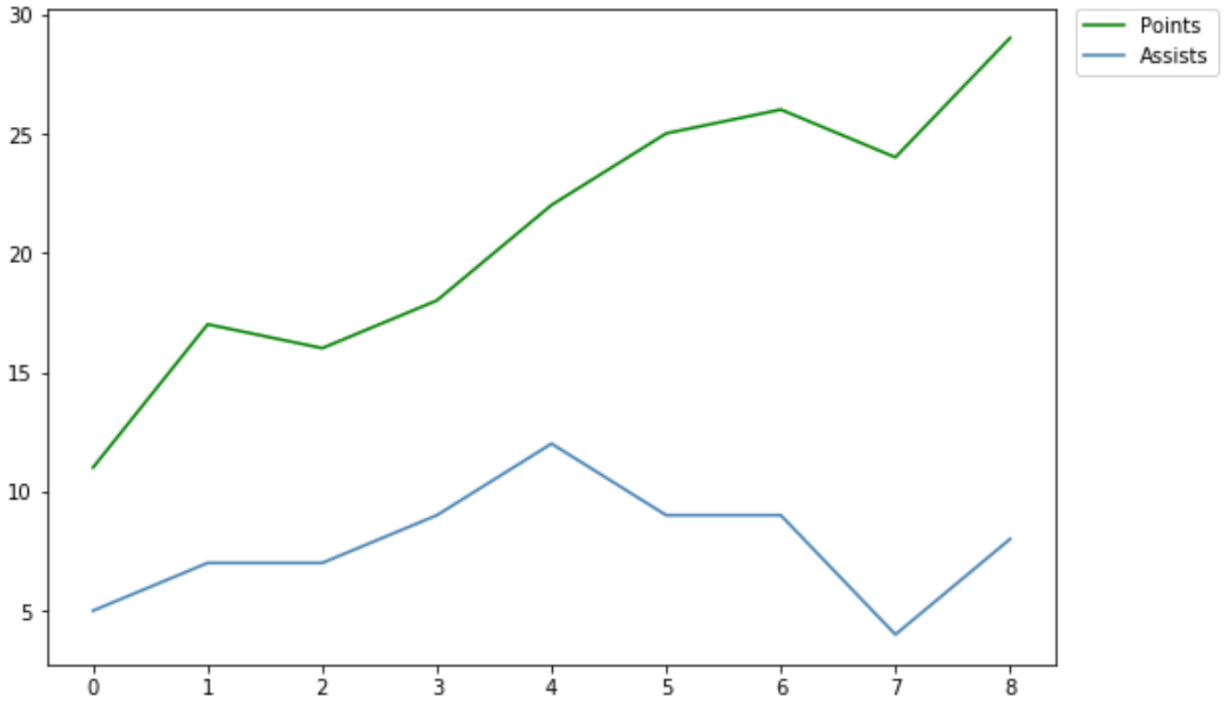
Here is how to place the legend just outside the top right corner of the plot area. We set the anchor point slightly beyond the right edge (1.02) and at the top edge (1), instructing the legend's upper left corner to align there:

```
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(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0)
```



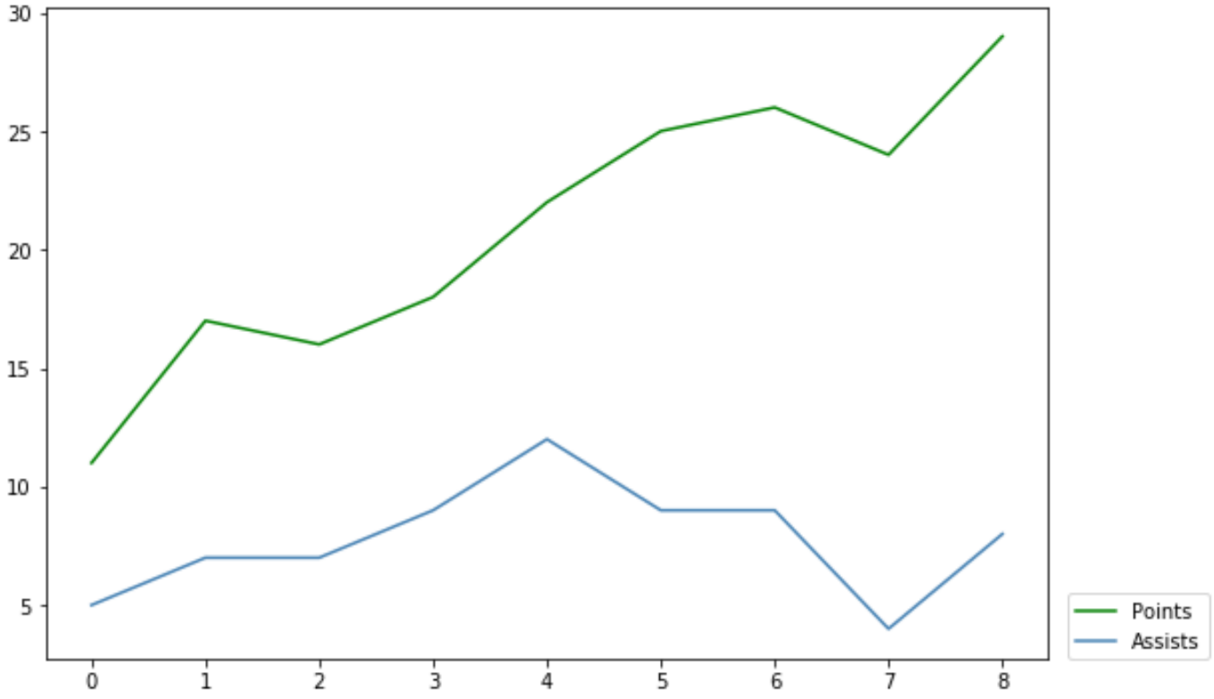
To illustrate the flexibility of the normalized coordinate system, we can easily shift the legend to the bottom right corner outside the plot by changing the y-coordinate of the `bbox_to_anchor()` tuple. By setting the y-coordinate to a small value (0.1), we anchor the legend near the bottom edge of 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 in center right of plot
plt.legend(bbox_to_anchor=(1.02, 0.1), loc='upper left', borderaxespad=0)



Refer to the official Matplotlib documentation for a detailed explanation of the **bbox_to_anchor()** argument and its application in advanced plotting scenarios.

Additional Resources

The following tutorials explain how to perform other common operations in Matplotlib, enhancing your proficiency in creating informative visualizations:

How to Add a Title to Legend in Matplotlib