**1. Importing Required Libraries**

import matplotlib.pyplot as plt

import numpy as np

from matplotlib.colors import ListedColormap

* matplotlib.pyplot is imported as plt. This module provides functions for plotting.
* numpy is imported as np. This module is used for handling arrays and numerical computations.
* ListedColormap is imported from matplotlib.colors. This is used to create a custom colormap.

**2. Generating Random Data**

x = np.random.rand(50)

y = np.random.rand(50)

colors = np.random.rand(50)

* np.random.rand(50) generates an array of 50 random floating-point numbers between 0 and 1.
* x and y contain the coordinates for 50 points.
* colors contains 50 random values that will be used to determine the color of each point.

Example values:

x = [0.13, 0.72, 0.25, ..., 0.89] # 50 values

y = [0.58, 0.33, 0.90, ..., 0.12] # 50 values

colors = [0.45, 0.12, 0.88, ..., 0.67] # 50 values

**3. Creating a Custom Colormap**

cmap = ListedColormap(['red', 'green', 'blue'])

* ListedColormap(['red', 'green', 'blue']) creates a colormap with three distinct colors: red, green, and blue.
* However, the colors assigned to each point will not strictly be red, green, or blue because colors contains continuous values from 0 to 1.
* Instead, the colormap interpolates between these three colors.

**4. Creating the Scatter Plot**

plt.scatter(x, y, c=colors, cmap=cmap)

* plt.scatter() creates a scatter plot.
* x, y are the coordinates of 50 points.
* c=colors assigns a color to each point based on the colors array.
* cmap=cmap applies the custom colormap (red, green, blue) to the color values.

Since colors are random values from 0 to 1, **Matplotlib maps them to the custom colormap**. The colors will **blend** between red, green, and blue depending on their values.

**5. Adding a Colorbar**

plt.colorbar()

* Adds a colorbar to the plot.
* The colorbar shows the mapping between numeric values in colors and the corresponding colors in cmap.

**6. Displaying the Plot**

CopyEdit

plt.show()

* plt.show() renders and displays the scatter plot.

**Summary of the Output**

* A scatter plot with 50 randomly placed points.
* Points are colored based on a custom colormap of red, green, and blue.
* A colorbar is displayed to show the mapping between numeric values and colors.

**# Customizing Colormaps**

**#You can modify existing colormaps by changing properties like color limits.**

Import numpy as np

from matplotlib import pyplot as plt

import matplotlib as mpl

* numpy is imported as np – used for handling arrays and numerical computations.
* pyplot from matplotlib is imported as plt – used for creating plots.
* matplotlib is imported as mpl – used to modify global Matplotlib settings.

**2. Generating Random Data**

data = np.random.rand(4, 4)

* np.random.rand(4, 4) creates a **4×4 matrix** of random values between **0 and 1**.
* Example output:

[[0.12, 0.87, 0.45, 0.23],

[0.76, 0.44, 0.98, 0.31],

[0.11, 0.55, 0.67, 0.89],

[0.23, 0.78, 0.09, 0.54]]

* This matrix will be used for visualization.

**3. Creating Subplots**

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(7, 4))

* plt.subplots(1, 2, figsize=(7, 4))
  + Creates **one row** and **two columns** of subplots.
  + figsize=(7, 4) sets the figure size to **7 inches wide** and **4 inches tall**.
* (ax1, ax2) are the two subplot axes.

**4. Displaying the Data Using imshow()**

ax1.imshow(data)

ax1.set\_title("Default colormap")

* ax1.imshow(data) displays the **4×4 matrix** as an image.
* By default, Matplotlib applies the default **colormap** (viridis in newer versions).
* ax1.set\_title("Default colormap") sets the title for the first subplot.

**5. Changing the Default Colormap**

mpl.rc('image', cmap='plasma')

* mpl.rc('image', cmap='plasma') **modifies the global default colormap** to **"plasma"**.
* This means that all future plots using imshow() (without specifying cmap) will use the "plasma" colormap.

**6. Displaying Data Again with New Colormap**

ax2.imshow(data)

ax2.set\_title("Modified colormap")

* ax2.imshow(data) displays the **same matrix** again.
* Since we changed the global default colormap to "plasma", this image will be **colored differently** from ax1.
* ax2.set\_title("Modified colormap") sets the title for the second subplot.

**7. Displaying the Figure**

plt.show()

* plt.show() renders and displays the plots.

**Summary of the Output**

* **First subplot (ax1)**: Uses the default colormap (viridis or jet depending on the Matplotlib version).
* **Second subplot (ax2)**: Uses the globally modified **"plasma"** colormap.
* The two subplots allow us to compare the effect of changing the colormap.