Matplotlib

Kittikun Jitpairod

Introduction to Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It provides a MATLAB-like interface, especially when used with IPython. Let's explore some common plot types and how to create them.

First, let's import the necessary libraries:

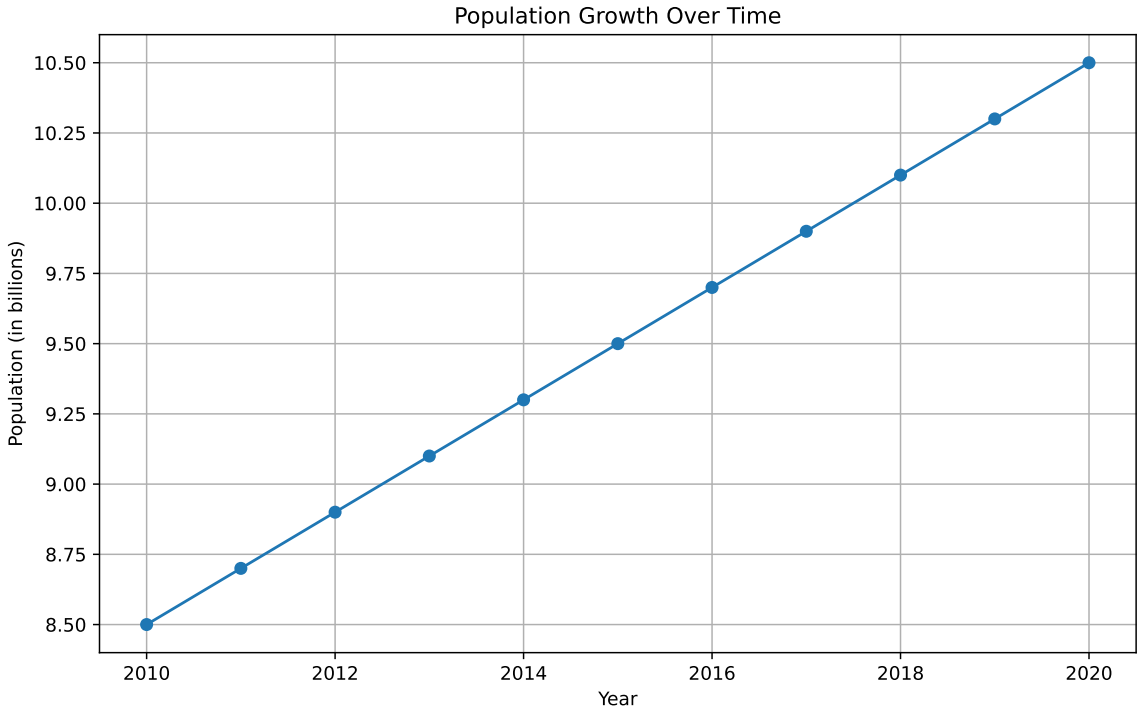
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

Line Plot

Line plots are great for showing trends over time or any continuous data.

```
years = np.arange(2010, 2021)
population = [8.5, 8.7, 8.9, 9.1,
              9.3, 9.5, 9.7, 9.9,
              10.1, 10.3, 10.5]

plt.figure(figsize=(10, 6))
plt.plot(years, population, marker='o')
plt.title('Population Growth Over Time')
plt.xlabel('Year')
plt.ylabel('Population (in billions)')
plt.grid(True)
plt.show()
```

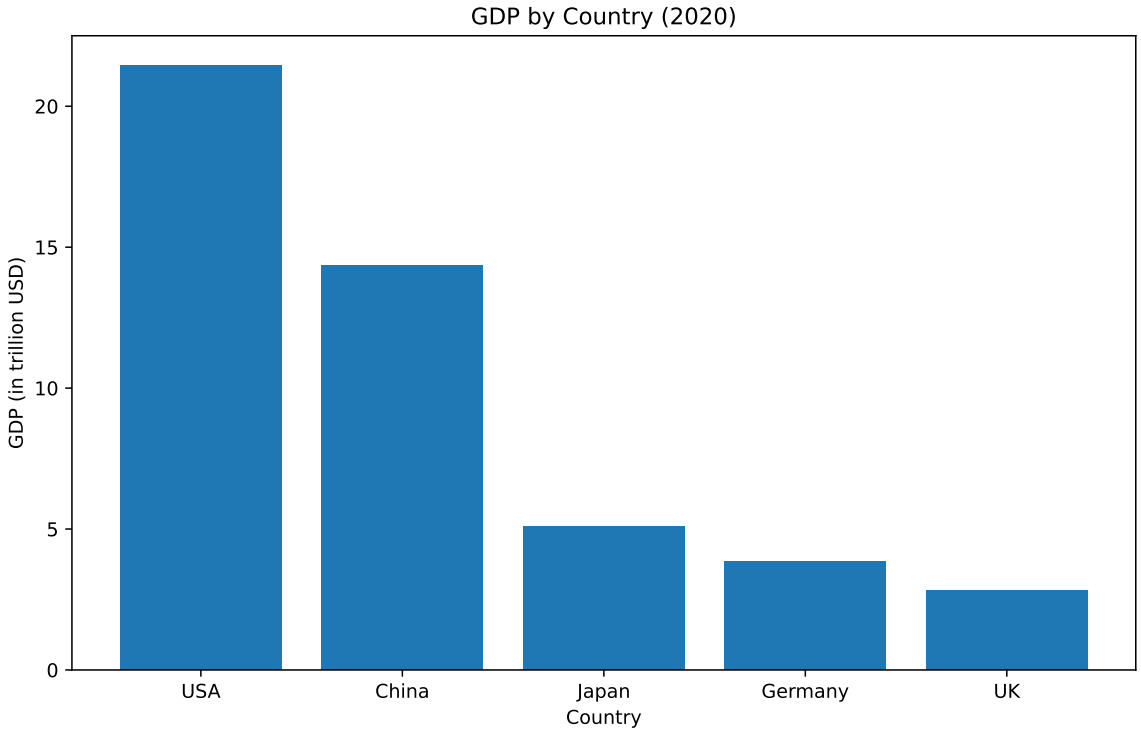


Vertical Bar Plot

Bar plots are useful for comparing quantities across different categories.

```
countries = ['USA', 'China', 'Japan',
             'Germany', 'UK']
gdp = [21.43, 14.34, 5.08, 3.86, 2.83]

plt.figure(figsize=(10, 6))
plt.bar(countries, gdp)
plt.title('GDP by Country (2020)')
plt.xlabel('Country')
plt.ylabel('GDP (in trillion USD)')
plt.show()
```



Horizontal Bar Plot

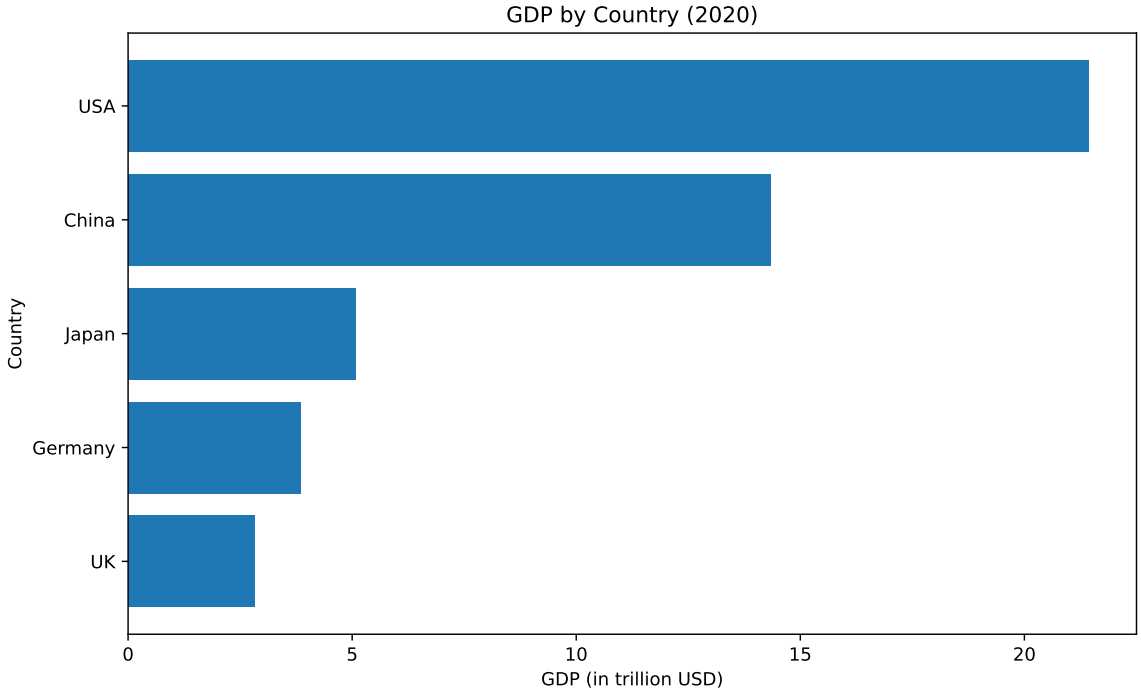
```
countries = ['China', 'Japan', 'USA',
             'Germany', 'UK']
gdp = [14.34, 5.08, 21.43, 3.86, 2.83]

df = pd.DataFrame({
    'Country': countries,
    'GDP': gdp,
})
df.sort_values(by = 'GDP', inplace = True)
print(df)

plt.figure(figsize=(10, 6))
plt.barh(df['Country'], df['GDP'])
plt.title('GDP by Country (2020)')
plt.xlabel('GDP (in trillion USD)')
plt.ylabel('Country')
plt.show()
```

```
   Country    GDP
4       UK   2.83
```

```
3   Germany   3.86
1     Japan   5.08
0     China  14.34
2       USA  21.43
```

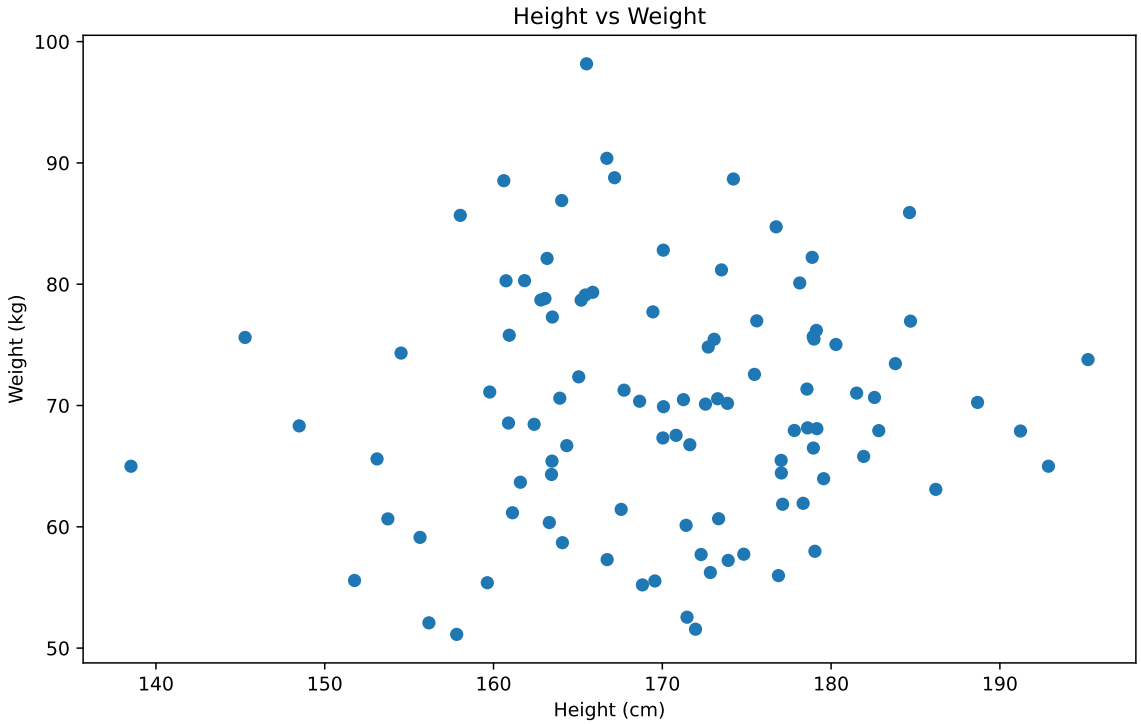


Scatter Plot

Scatter plots are excellent for showing the relationship between two variables.

```
height = np.random.normal(170, 10, 100)
weight = np.random.normal(70, 10, 100)

plt.figure(figsize=(10, 6))
plt.scatter(height, weight)
plt.title('Height vs Weight')
plt.xlabel('Height (cm)')
plt.ylabel('Weight (kg)')
plt.show()
```

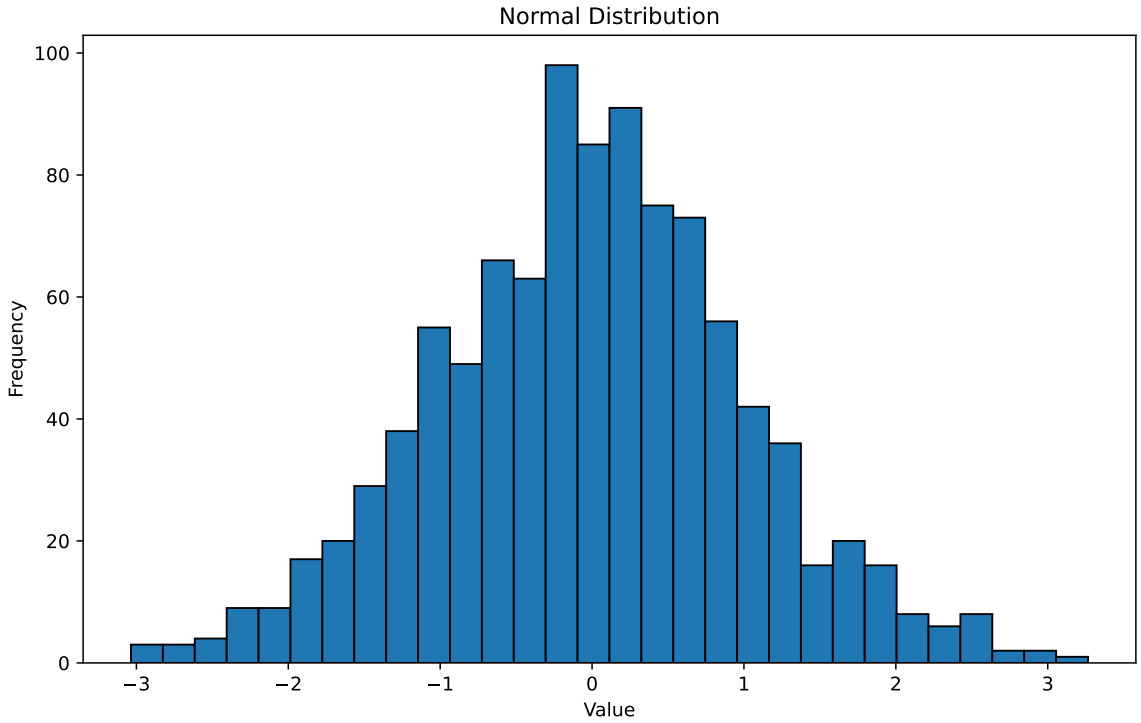


Histogram

Histograms show the distribution of a dataset.

```
data = np.random.normal(0, 1, 1000)

plt.figure(figsize=(10, 6))
plt.hist(data, bins=30, edgecolor='black')
plt.title('Normal Distribution')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```

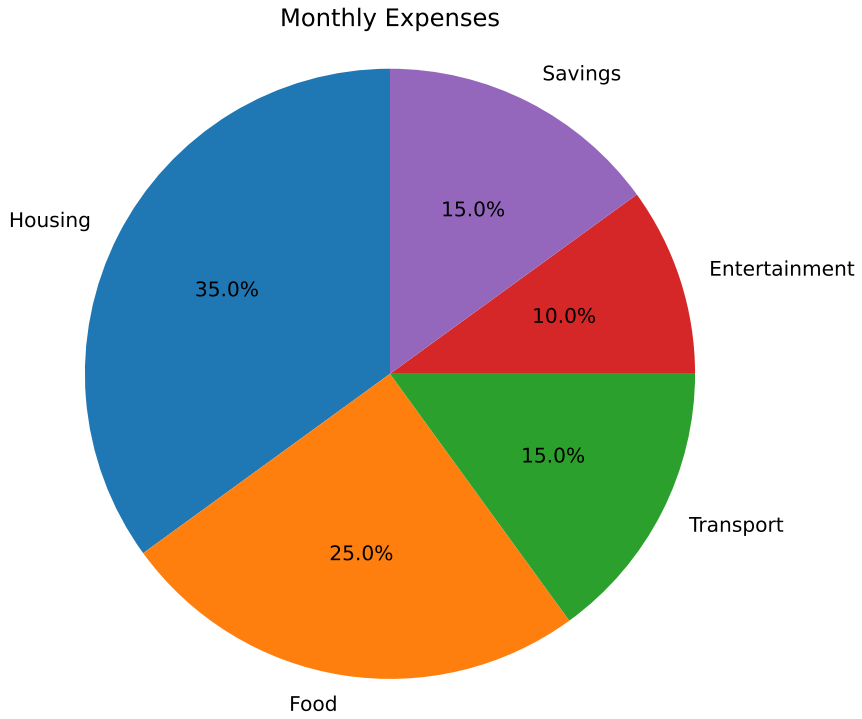


Pie Chart

Pie charts are used to show proportions of a whole.

```
categories = ['Housing', 'Food',
    'Transport', 'Entertainment',
    'Savings']
expenses = [35, 25, 15, 10, 15]

plt.figure(figsize=(10, 6))
plt.pie(expenses, labels=categories,
    autopct='%1.1f%%', startangle=90)
plt.title('Monthly Expenses')
plt.axis('equal')  # Equal aspect ratio
plt.show()
```



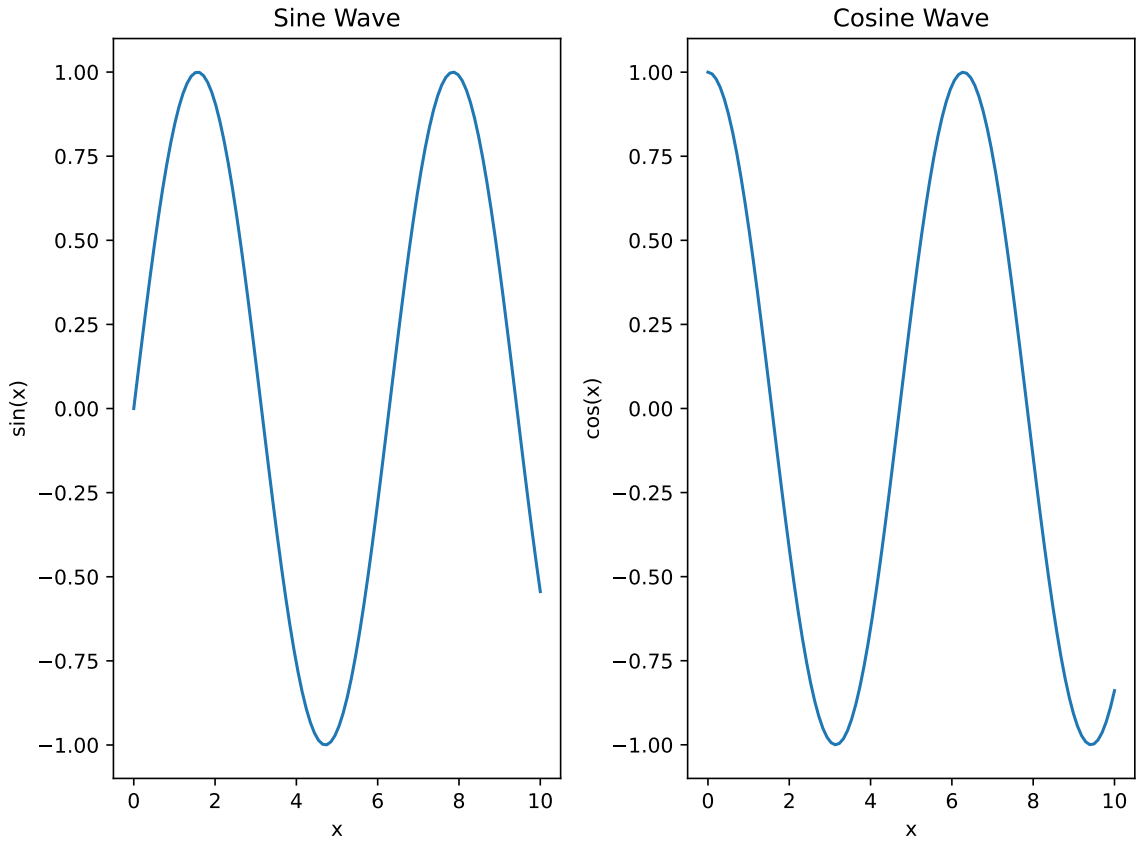
Subplots

Subplots allow you to combine multiple plots in one figure.

```
x = np.linspace(0, 10, 100)
y1 = np.sin(x)
y2 = np.cos(x)

fig, axs = plt.subplots(1,2,figsize=(8,6))

axs[0].plot(x, y1)
axs[0].set_title('Sine Wave')
axs[0].set_xlabel('x')
axs[0].set_ylabel('sin(x)')
axs[1].plot(x, y2)
axs[1].set_title('Cosine Wave')
axs[1].set_xlabel('x')
axs[1].set_ylabel('cos(x)')
plt.tight_layout()
plt.show()
```

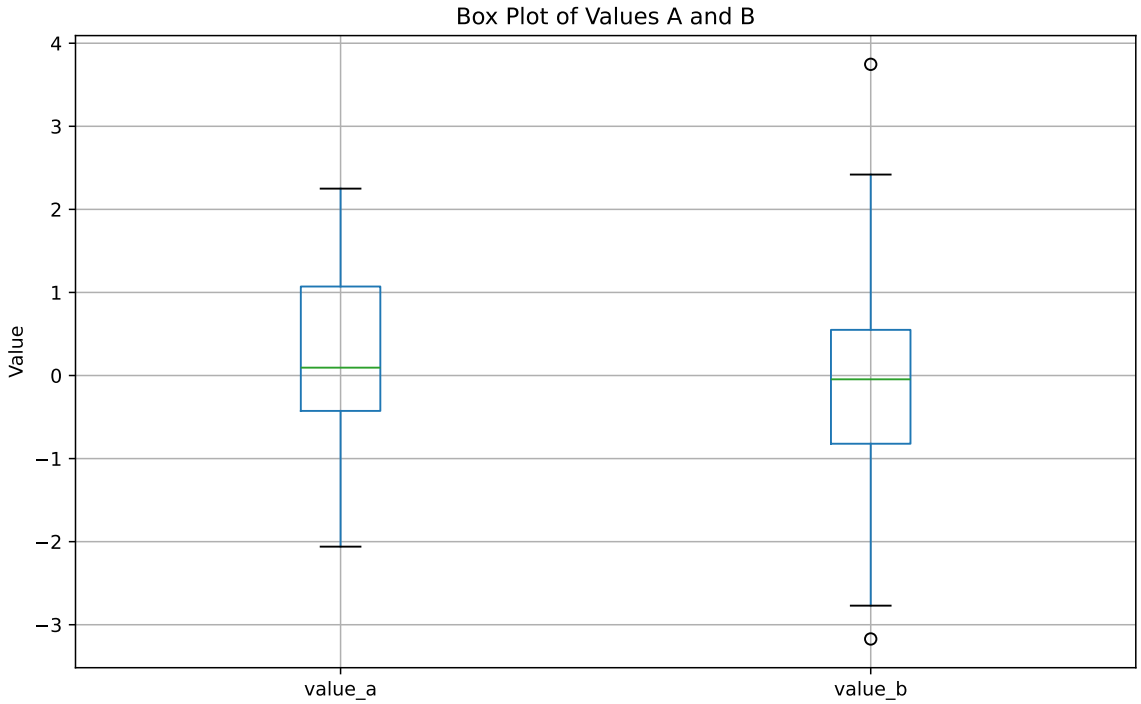


Box Plot

Box plots are useful for showing the distribution of data and identifying outliers:

```
df = pd.DataFrame({
    'value_a': np.random.randn(100),
    'value_b': np.random.randn(100),
})

df.boxplot(column=['value_a', 'value_b'],
    figsize=(10, 6))
plt.title('Box Plot of Values A and B')
plt.ylabel('Value')
plt.grid(True)
plt.show()
```

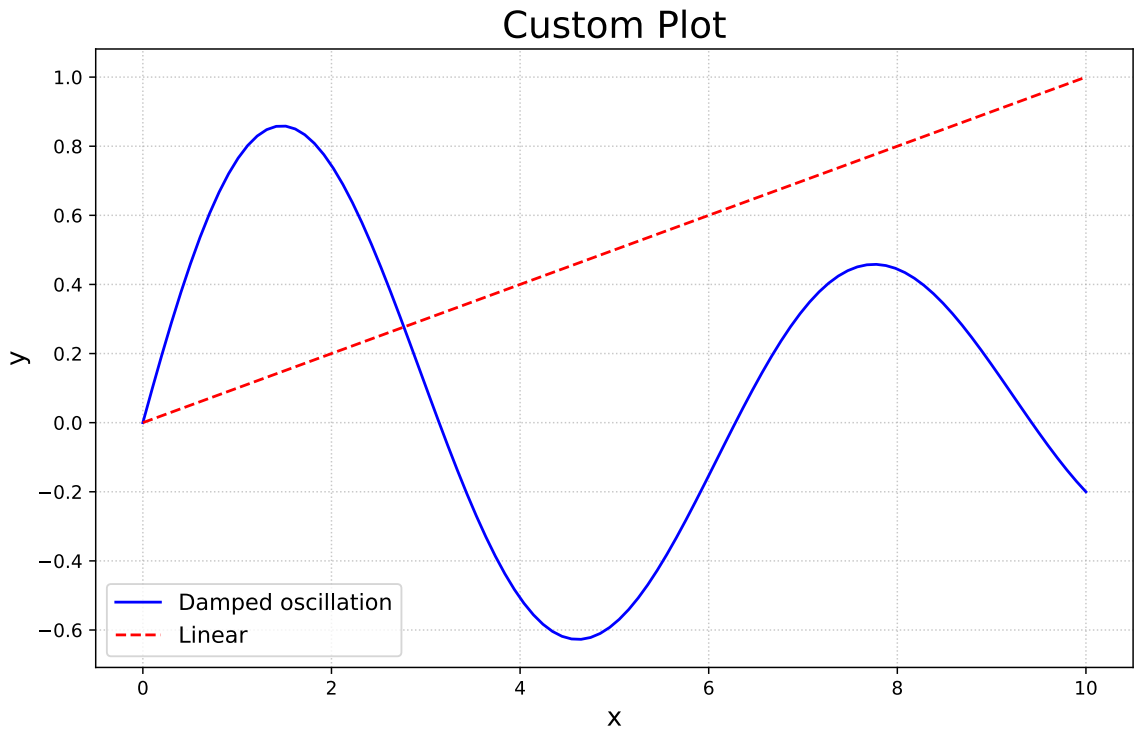


Customizing Plots

Matplotlib offers many customization options. Here's an example with a customized line plot:

```
x = np.linspace(0, 10, 100)
y1 = np.exp(-x/10)*np.sin(x)
y2 = x/10

plt.figure(figsize=(10, 6))
plt.plot(x, y1, 'b-',
    label='Damped oscillation')
plt.plot(x, y2, 'r--', label='Linear')
plt.title('Custom Plot', fontsize=20)
plt.xlabel('x', fontsize=14)
plt.ylabel('y', fontsize=14)
plt.legend(fontsize=12)
plt.grid(True, linestyle=':', alpha=0.7)
plt.show()
```


Custom Plot
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
y
0
2
4
6
8
10
x
Damped oscillation
Linear