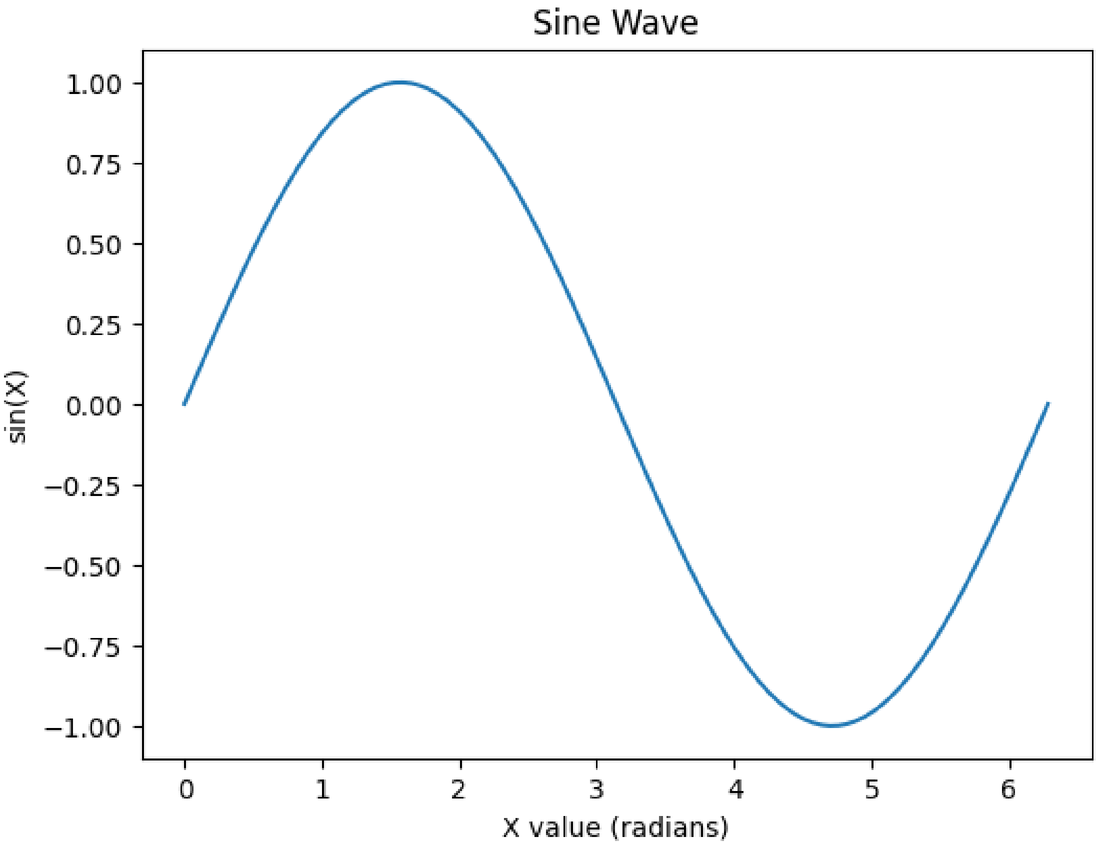
Matplotlib_Basics

June 8, 2025

0.1 Plotting a Basic Line Graph

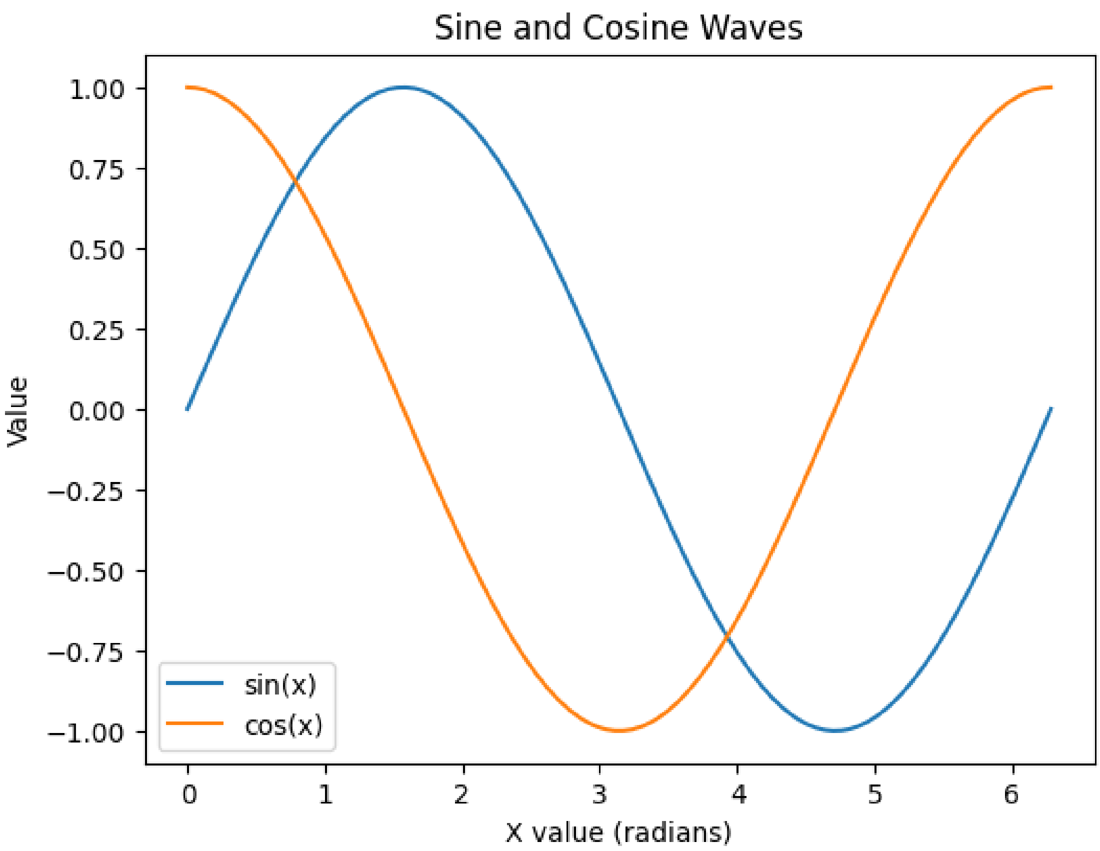
[4]:

```python
# Import the necessary modules
import numpy as np
import matplotlib.pyplot as plt

# Prepare data for a simple line plot: y = sin(x)
x = np.linspace(0, 2*np.pi, 100)   # 100 points from 0 to 2pi
y = np.sin(x)                      # Compute sine of each x

# Create a line plot
plt.plot(x, y)                     # Plot y versus x as a line
plt.title('Sine Wave')             # Add a title to the plot
plt.xlabel('X value (radians)')    # Label for x-axis
plt.ylabel('sin(X)')               # Label for y-axis

plt.show()                         # Display the figure
```



0.2 Plotting Multiple Lines and Adding a Legend

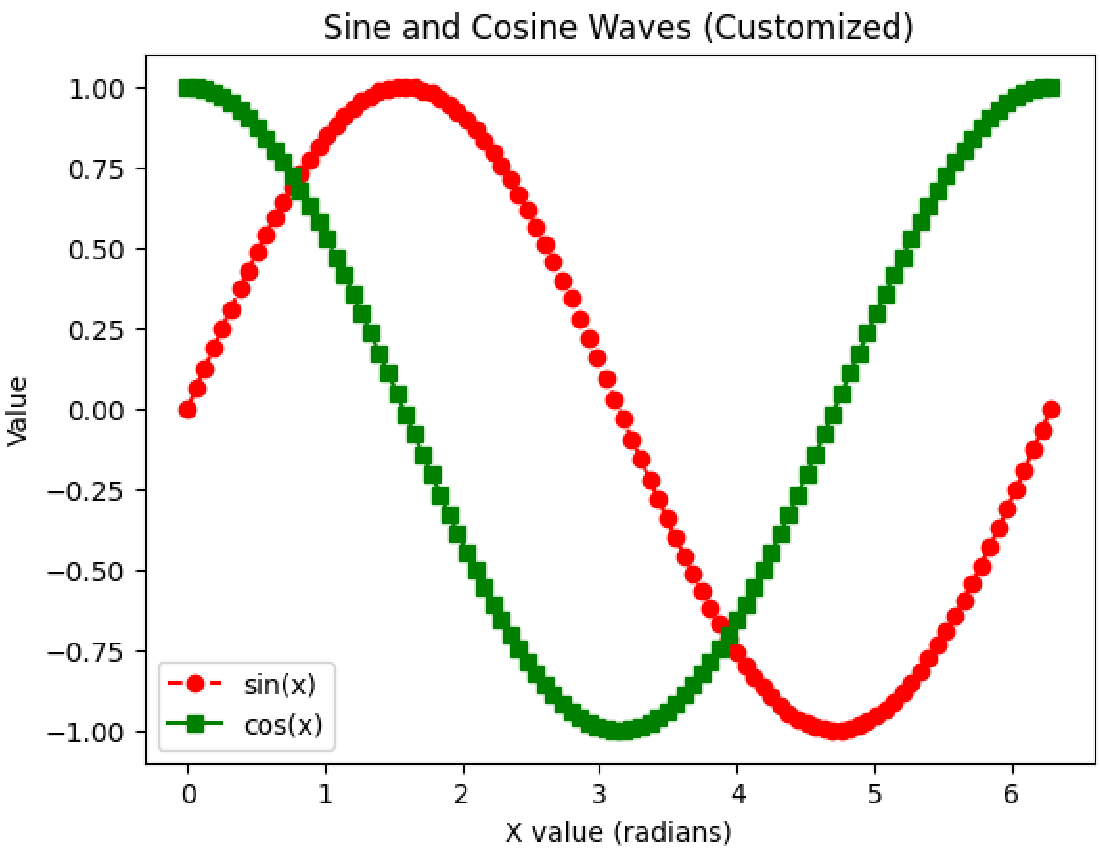
[5]:
```
# Plotting multiple lines on the same graph
y2 = np.cos(x)  # Another dataset: cosine wave

plt.plot(x, y, label='sin(x)')    # first line
plt.plot(x, y2, label='cos(x)')   # second line
plt.title('Sine and Cosine Waves')
plt.xlabel('X value (radians)')
plt.ylabel('Value')

plt.legend()   # display legend to show labels for each line
plt.show()
```



0.3 Customizing Colors, Linestyles, and Markers

[6]:

```
# Customize line styles, colors, and markers
plt.plot(x, y, color='red', linestyle='--', marker='o', label='sin(x)')   # red␣
 ↪dashed line with circle markers
plt.plot(x, y2, color='green', linestyle='-', marker='s', label='cos(x)') #␣
 ↪green solid line with square markers

plt.title('Sine and Cosine Waves (Customized)')
plt.xlabel('X value (radians)')
plt.ylabel('Value')
plt.legend()

plt.show()
```

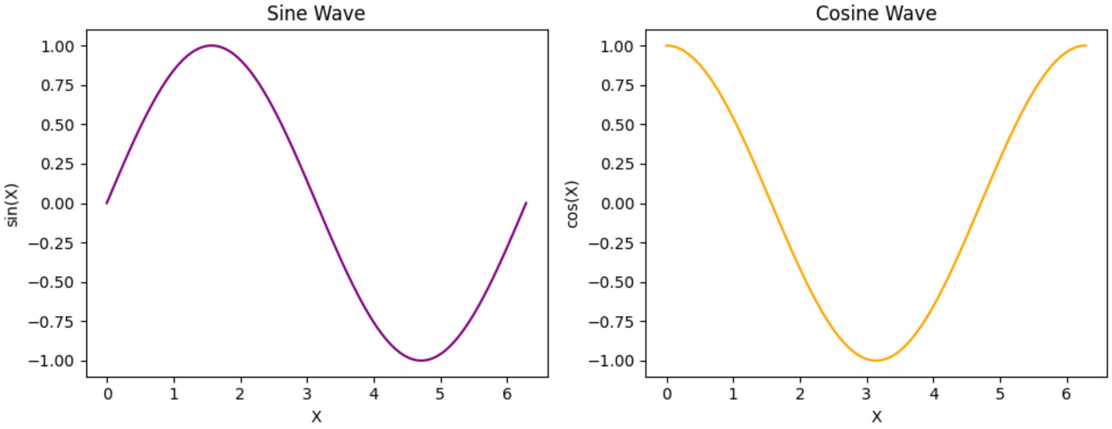


0.4 Subplots: Multiple Plots in One Figure

[7]:

```
# Create a figure with two subplots (1 row, 2 columns)
fig, axes = plt.subplots(1, 2, figsize=(10,4))  # 1 row, 2 columns, figure size␣
 ↪10x4 inches

# axes is an array of Axes objects since we have 2 subplots
ax1, ax2 = axes  # unpack the axes for clarity

# Plot sine wave on the first subplot
ax1.plot(x, y, color='purple')
ax1.set_title('Sine Wave')
ax1.set_xlabel('X')
ax1.set_ylabel('sin(X)')

# Plot cosine wave on the second subplot
ax2.plot(x, y2, color='orange')
ax2.set_title('Cosine Wave')
ax2.set_xlabel('X')
```

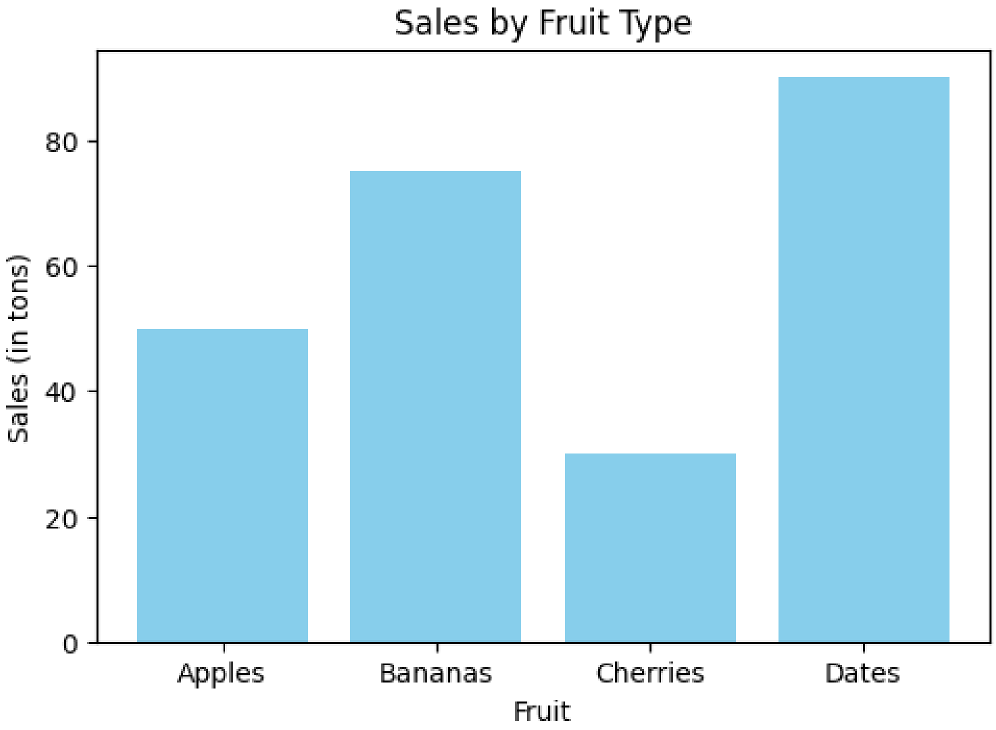
```
ax2.set_ylabel('cos(X)')

plt.tight_layout()  # adjust subplot spacing to fit nicely in the figure
plt.show()
```



0.5 Bar Charts

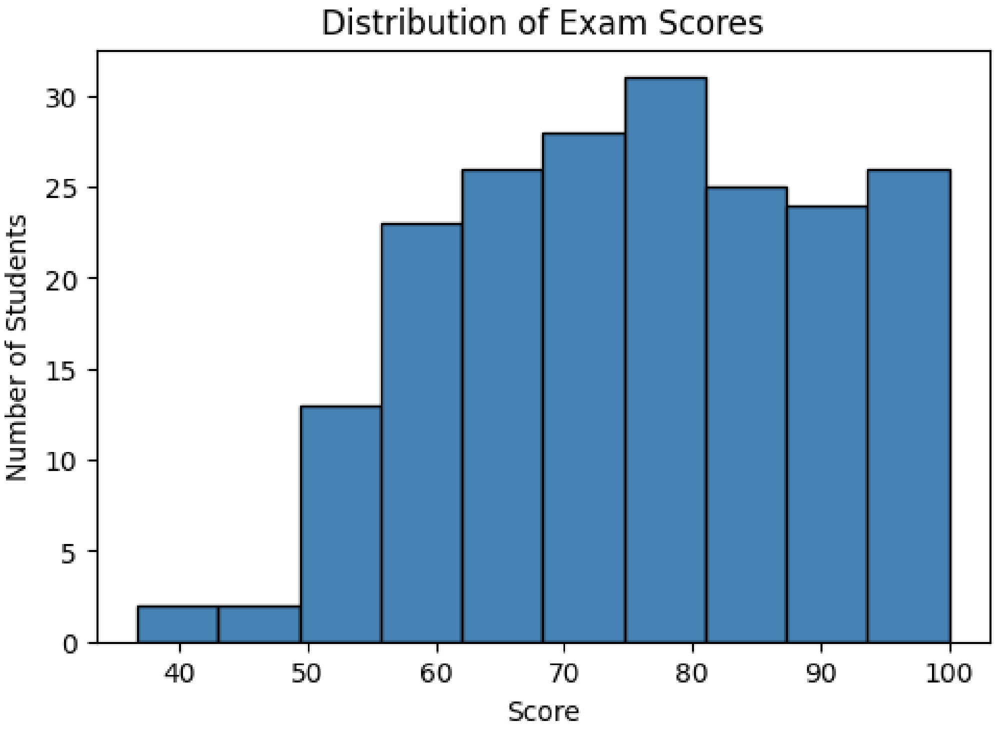
[8]:

```
# Sample categorical data for bar chart
categories = ['Apples', 'Bananas', 'Cherries', 'Dates']
values = [50, 75, 30, 90]  # e.g., sales figures for each category

# Create a bar chart
plt.figure(figsize=(6,4))
plt.bar(categories, values, color='skyblue')  # bar chart with custom color

plt.title('Sales by Fruit Type')
plt.xlabel('Fruit')
plt.ylabel('Sales (in tons)')

plt.show()
```



0.6 Histograms

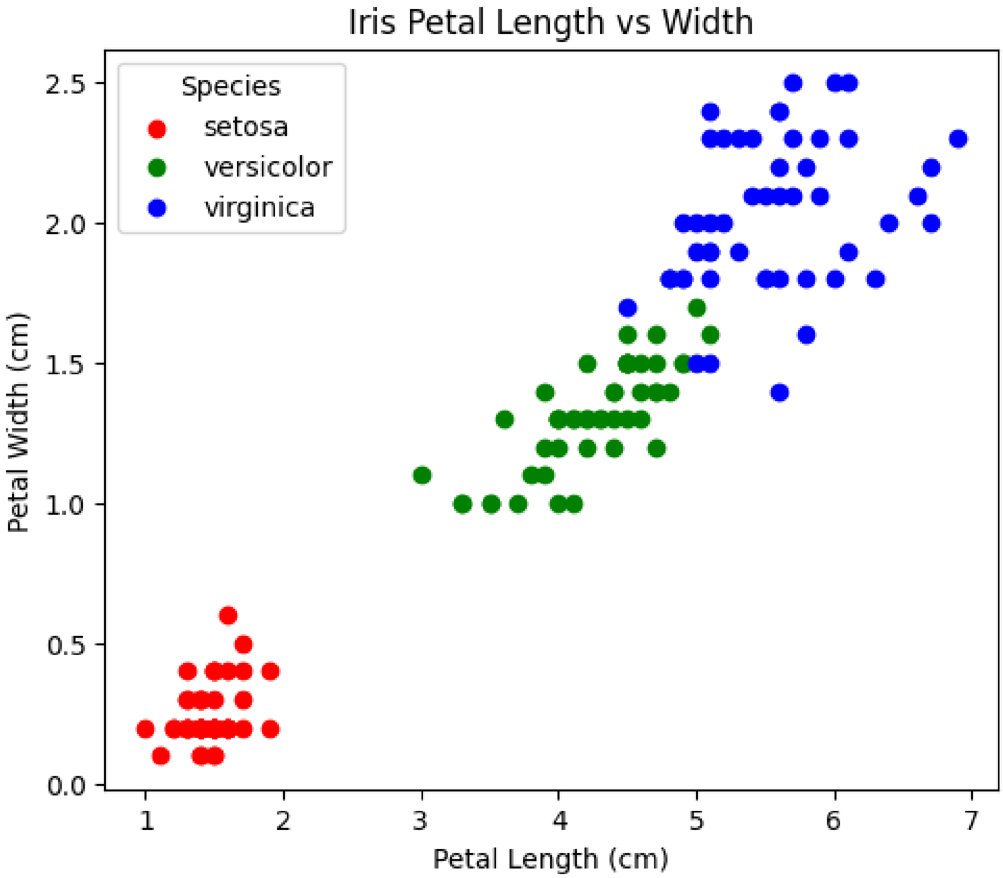
[9]:
```
# Generate random data for a histogram (e.g., exam scores out of 100)
np.random.seed(0)  # for reproducibility
scores = np.random.normal(loc=75, scale=15, size=200)  # mean 75, std 15
scores = np.clip(scores, 0, 100)  # ensure scores are between 0 and 100

plt.figure(figsize=(6,4))
plt.hist(scores, bins=10, color='steelblue', edgecolor='black')  # histogram␣
 ↪with 10 bins
plt.title('Distribution of Exam Scores')
plt.xlabel('Score')
plt.ylabel('Number of Students')

plt.show()
```



0.7 Scatter Plots

[10]:

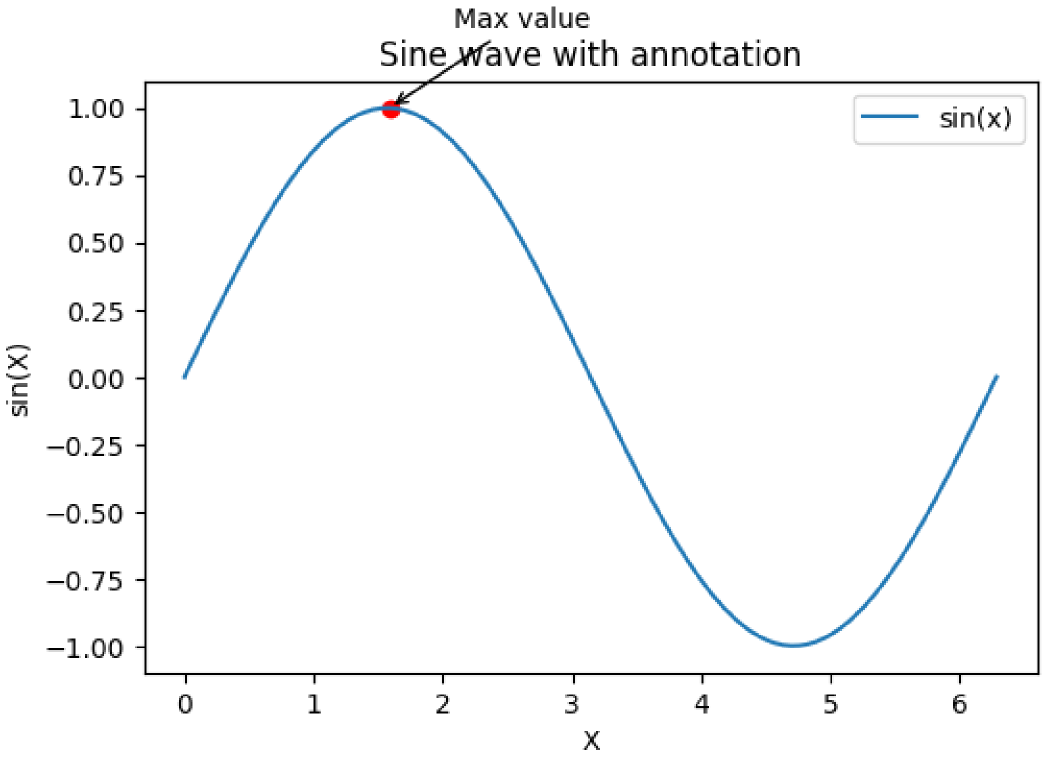
```
from sklearn.datasets import load_iris
import pandas as pd

# Load from sklearn
data = load_iris(as_frame=True)
iris = data.frame
iris['species'] = iris['target'].apply(lambda i: data.target_names[i])

# Now use the same plotting logic
import matplotlib.pyplot as plt

plt.figure(figsize=(6,5))
species_unique = iris['species'].unique()
colors = {'setosa':'red', 'versicolor':'green', 'virginica':'blue'}

for sp in species_unique:
    subset = iris[iris['species'] == sp]
    plt.scatter(subset['petal length (cm)'], subset['petal width (cm)'],␣
  ↪color=colors[sp], label=sp)

plt.title('Iris Petal Length vs Width')
plt.xlabel('Petal Length (cm)')
plt.ylabel('Petal Width (cm)')
plt.legend(title='Species')
plt.show()
```



0.8 Adding Text Annotations

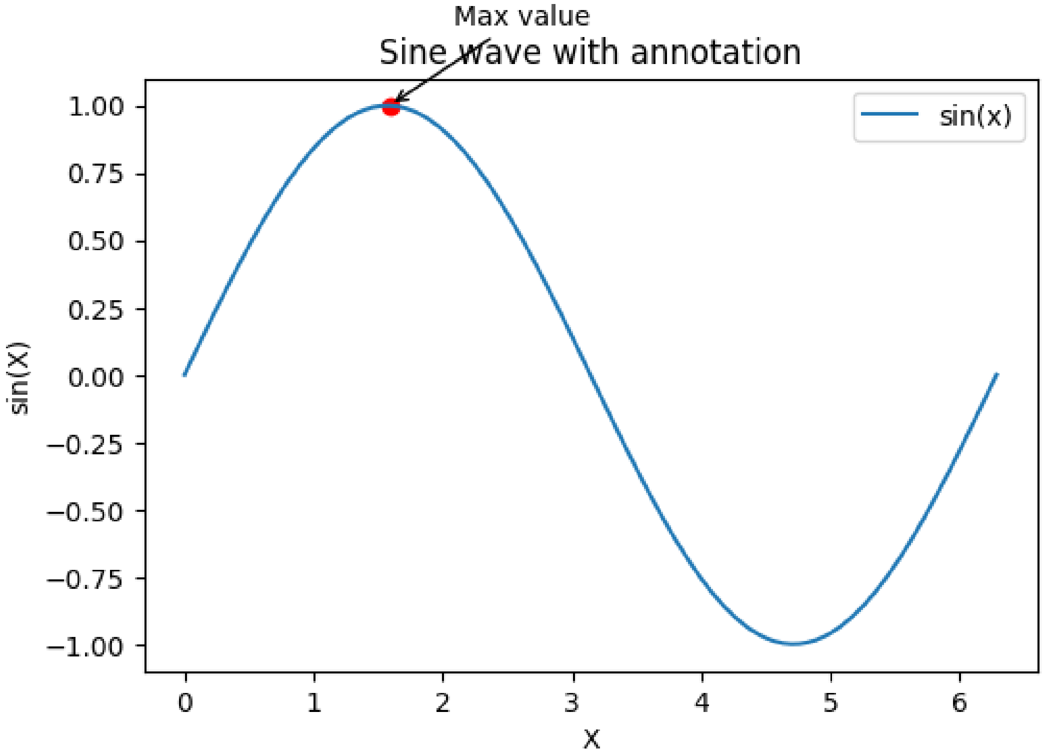
[11]:
```
import numpy as np
```

[12]:
```
# Annotating the maximum point of a sine wave
x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)
max_idx = np.argmax(y)             # index of maximum y value
x_max = x[max_idx]
y_max = y[max_idx]

plt.figure(figsize=(6,4))
plt.plot(x, y, label='sin(x)')
plt.title('Sine wave with annotation')
plt.xlabel('X')
plt.ylabel('sin(X)')

# Annotate the maximum point
plt.scatter(x_max, y_max, color='red')  # draw a red dot at the max point
plt.annotate('Max value', xy=(x_max, y_max), xytext=(x_max+0.5, y_max+0.3),
             arrowprops=dict(facecolor='black', arrowstyle='->'))

plt.legend()
plt.show()
```



0.9 Adding Text Annotations

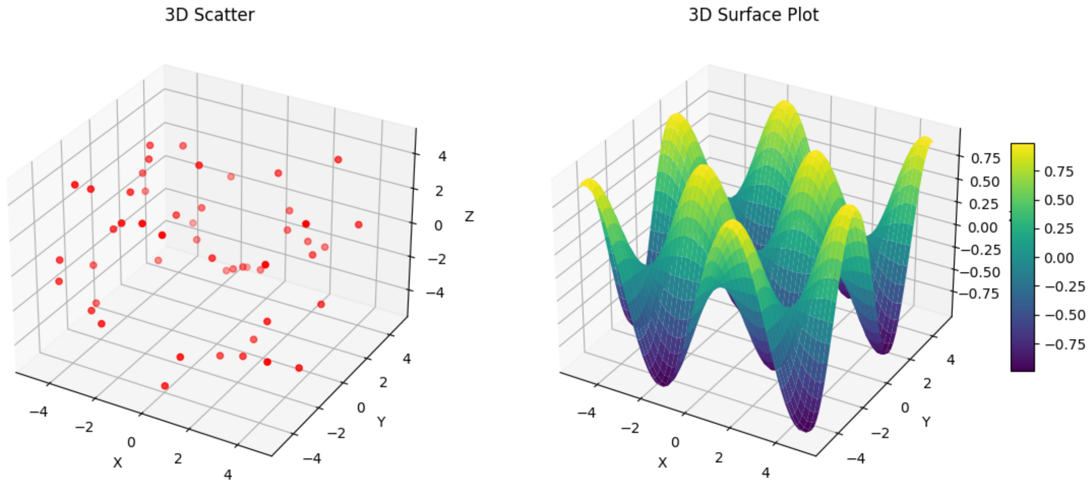
[13]:
```
# Annotating the maximum point of a sine wave
x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)
max_idx = np.argmax(y)            # index of maximum y value
x_max = x[max_idx]
y_max = y[max_idx]

plt.figure(figsize=(6,4))
plt.plot(x, y, label='sin(x)')
plt.title('Sine wave with annotation')
plt.xlabel('X')
plt.ylabel('sin(X)')

# Annotate the maximum point
plt.scatter(x_max, y_max, color='red')  # draw a red dot at the max point
plt.annotate('Max value', xy=(x_max, y_max), xytext=(x_max+0.5, y_max+0.3),
             arrowprops=dict(facecolor='black', arrowstyle='->'))

plt.legend()
plt.show()
```



0.10 3D Plots

[14]:
```
# 3D scatter and surface plot example
from mpl_toolkits.mplot3d import Axes3D  # though not strictly needed in newer␣
 ↪matplotlib versions

# Prepare data for 3D scatter
np.random.seed(42)
xs = np.random.uniform(-5, 5, 50)
ys = np.random.uniform(-5, 5, 50)
```

```
zs = np.random.uniform(-5, 5, 50)

# Prepare data for 3D surface (a mathematical function)
x_lin = np.linspace(-5, 5, 50)
y_lin = np.linspace(-5, 5, 50)
X, Y = np.meshgrid(x_lin, y_lin)
Z = np.sin(X) * np.sin(Y)  # just an example function

# Create subplots: one for scatter, one for surface
fig = plt.figure(figsize=(12,5))
ax1 = fig.add_subplot(1, 2, 1, projection='3d')
ax2 = fig.add_subplot(1, 2, 2, projection='3d')

# 3D Scatter plot
ax1.scatter(xs, ys, zs, color='red', marker='o')
ax1.set_title('3D Scatter')
ax1.set_xlabel('X')
ax1.set_ylabel('Y')
ax1.set_zlabel('Z')

# 3D Surface plot
surf = ax2.plot_surface(X, Y, Z, cmap='viridis')
ax2.set_title('3D Surface Plot')
ax2.set_xlabel('X')
ax2.set_ylabel('Y')
ax2.set_zlabel('Z')
# Add a color bar for the surface plot to show the color scale
fig.colorbar(surf, ax=ax2, shrink=0.5, aspect=10)

plt.tight_layout()
plt.show()
```



0.11 Interactive Widgets with ipywidgets

[17]:

```
!pip install ipywidgets
```

```
Collecting ipywidgets
  Downloading ipywidgets-8.1.7-py3-none-any.whl.metadata (2.4 kB)
Requirement already satisfied: comm>=0.1.3 in
c:\users\isha\anaconda3\envs\vizfix\lib\site-packages (from ipywidgets) (0.2.1)
Requirement already satisfied: ipython>=6.1.0 in
c:\users\isha\anaconda3\envs\vizfix\lib\site-packages (from ipywidgets) (8.30.0)
Requirement already satisfied: traitlets>=4.3.1 in
c:\users\isha\anaconda3\envs\vizfix\lib\site-packages (from ipywidgets) (5.14.3)
Collecting widgetsnbextension~=4.0.14 (from ipywidgets)
  Downloading widgetsnbextension-4.0.14-py3-none-any.whl.metadata (1.6 kB)
Collecting jupyterlab_widgets~=3.0.15 (from ipywidgets)
  Downloading jupyterlab_widgets-3.0.15-py3-none-any.whl.metadata (20 kB)
Requirement already satisfied: decorator in
c:\users\isha\anaconda3\envs\vizfix\lib\site-packages (from
ipython>=6.1.0->ipywidgets) (5.1.1)
Requirement already satisfied: jedi>=0.16 in
c:\users\isha\anaconda3\envs\vizfix\lib\site-packages (from
ipython>=6.1.0->ipywidgets) (0.19.2)
Requirement already satisfied: matplotlib-inline in
c:\users\isha\anaconda3\envs\vizfix\lib\site-packages (from
ipython>=6.1.0->ipywidgets) (0.1.6)
Requirement already satisfied: prompt-toolkit<3.1.0,>=3.0.41 in
c:\users\isha\anaconda3\envs\vizfix\lib\site-packages (from
ipython>=6.1.0->ipywidgets) (3.0.43)
Requirement already satisfied: pygments>=2.4.0 in
c:\users\isha\anaconda3\envs\vizfix\lib\site-packages (from
ipython>=6.1.0->ipywidgets) (2.19.1)
Requirement already satisfied: stack-data in
c:\users\isha\anaconda3\envs\vizfix\lib\site-packages (from
ipython>=6.1.0->ipywidgets) (0.2.0)
Requirement already satisfied: exceptiongroup in
c:\users\isha\anaconda3\envs\vizfix\lib\site-packages (from
ipython>=6.1.0->ipywidgets) (1.2.0)
Requirement already satisfied: typing-extensions>=4.6 in
c:\users\isha\anaconda3\envs\vizfix\lib\site-packages (from
ipython>=6.1.0->ipywidgets) (4.12.2)
Requirement already satisfied: colorama in
c:\users\isha\anaconda3\envs\vizfix\lib\site-packages (from
ipython>=6.1.0->ipywidgets) (0.4.6)
Requirement already satisfied: wcwidth in
c:\users\isha\anaconda3\envs\vizfix\lib\site-packages (from prompt-
toolkit<3.1.0,>=3.0.41->ipython>=6.1.0->ipywidgets) (0.2.5)
```

```
Requirement already satisfied: parso<0.9.0,>=0.8.4 in
c:\users\isha\anaconda3\envs\vizfix\lib\site-packages (from
jedi>=0.16->ipython>=6.1.0->ipywidgets) (0.8.4)
Requirement already satisfied: executing in
c:\users\isha\anaconda3\envs\vizfix\lib\site-packages (from stack-
data->ipython>=6.1.0->ipywidgets) (0.8.3)
Requirement already satisfied: asttokens in
c:\users\isha\anaconda3\envs\vizfix\lib\site-packages (from stack-
data->ipython>=6.1.0->ipywidgets) (3.0.0)
Requirement already satisfied: pure-eval in
c:\users\isha\anaconda3\envs\vizfix\lib\site-packages (from stack-
data->ipython>=6.1.0->ipywidgets) (0.2.2)
Downloading ipywidgets-8.1.7-py3-none-any.whl (139 kB)
Downloading jupyterlab_widgets-3.0.15-py3-none-any.whl (216 kB)
Downloading widgetsnbextension-4.0.14-py3-none-any.whl (2.2 MB)
   ---------------------------------------- 2.2/2.2 MB 24.5 MB/s eta 0:00:00
Installing collected packages: widgetsnbextension, jupyterlab_widgets,
ipywidgets

Successfully installed ipywidgets-8.1.7 jupyterlab_widgets-3.0.15
widgetsnbextension-4.0.14
```

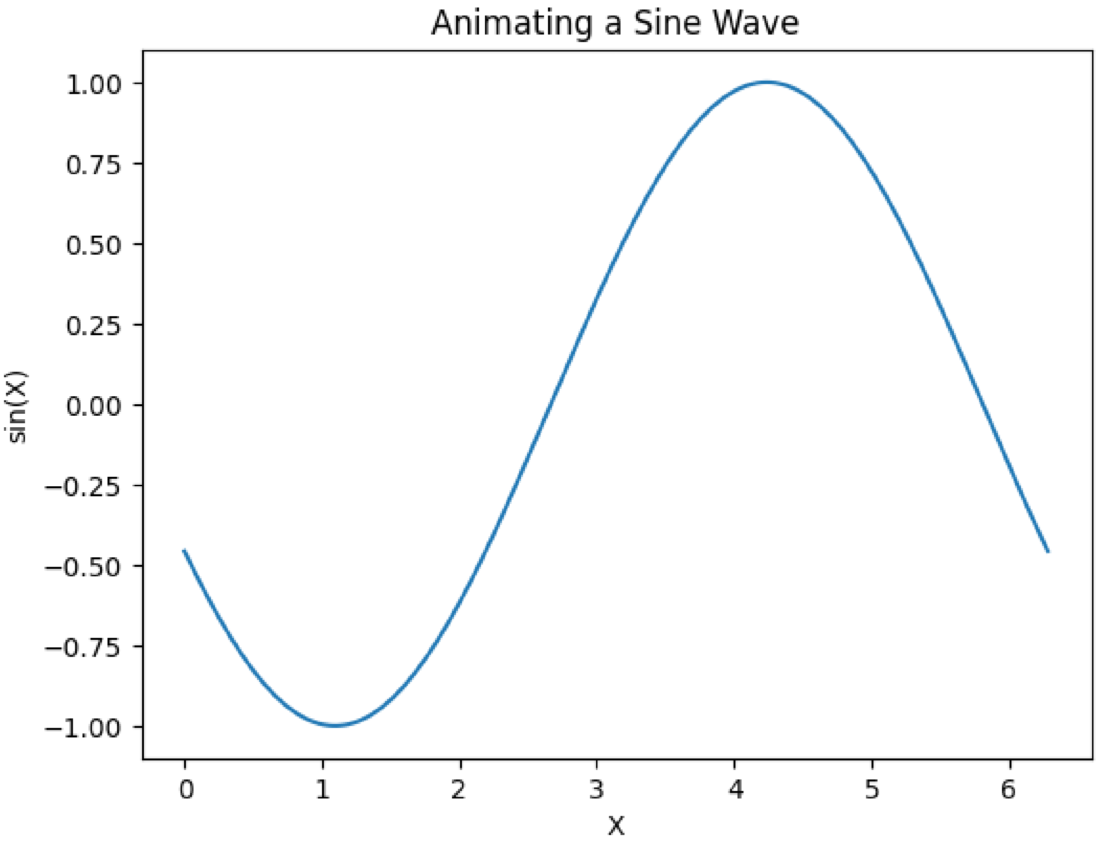
0.12 Animations

[21]:

```python
import matplotlib.animation as animation
from IPython.display import HTML

# Set up figure and axis
fig, ax = plt.subplots()
x = np.linspace(0, 2*np.pi, 200)
line, = ax.plot(x, np.sin(x))  # initial line (will update in animation)
ax.set_ylim(-1.1, 1.1)
ax.set_xlabel('X')
ax.set_ylabel('sin(X)')
ax.set_title('Animating a Sine Wave')

# Animation update function
def animate(frame):
    # Shift the sine wave by a phase proportional to frame
    y = np.sin(x + frame/10.0)
    line.set_ydata(y)
    return line,

# Create animation
anim = animation.FuncAnimation(fig, animate, frames=100, interval=50, blit=True)

# Display the animation in the notebook (as HTML)
```

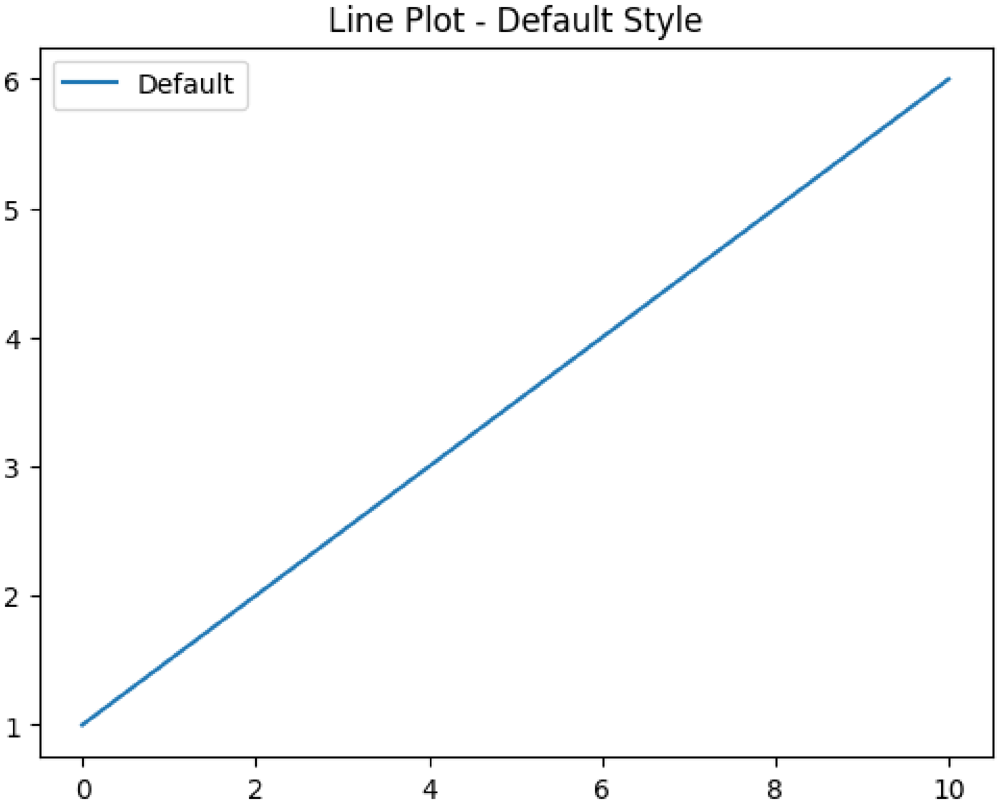
```
HTML(anim.to_jshtml())
```

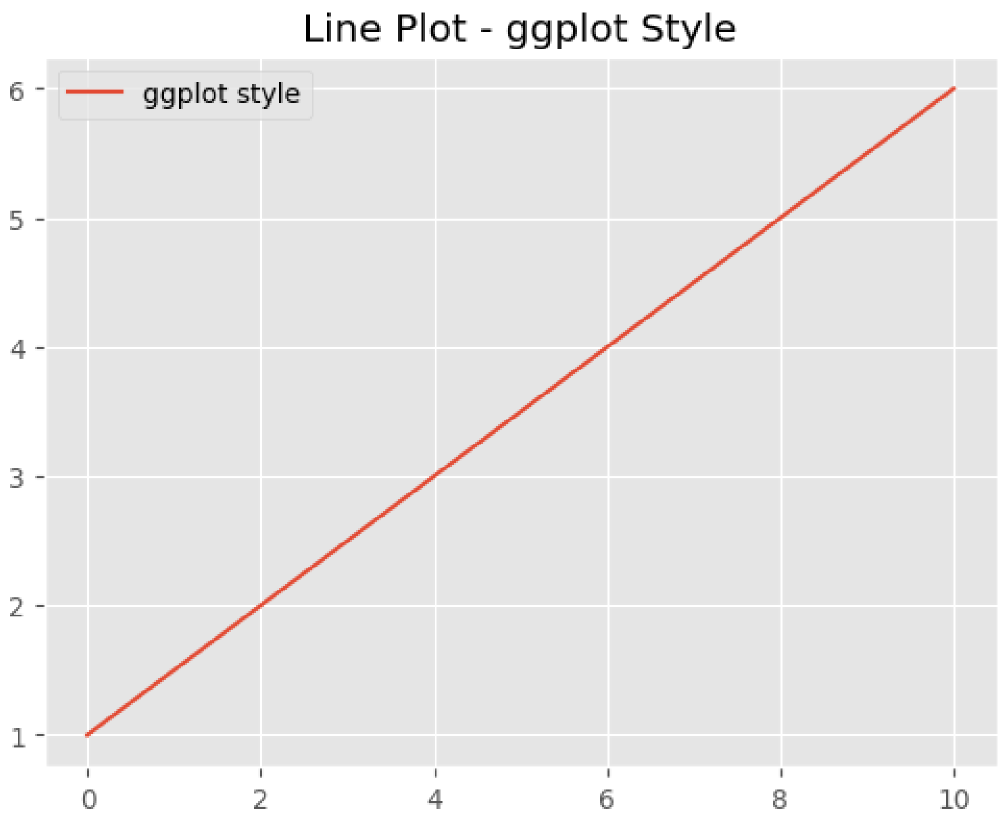
[21]: <IPython.core.display.HTML object>



0.13 Styling with Matplotlib Stylesheets

[16]:
```
# Plot with default style
x = np.linspace(0, 10, 100)
y = 0.5 * x + 1
plt.figure()
plt.plot(x, y, label='Default')
plt.title('Line Plot - Default Style')
plt.legend()
plt.show()

# Plot with ggplot style
plt.style.use('ggplot')
plt.figure()
plt.plot(x, y, label='ggplot style')
plt.title('Line Plot - ggplot Style')
plt.legend()
plt.show()

# Revert to default style for further plots
plt.style.use('default')
```

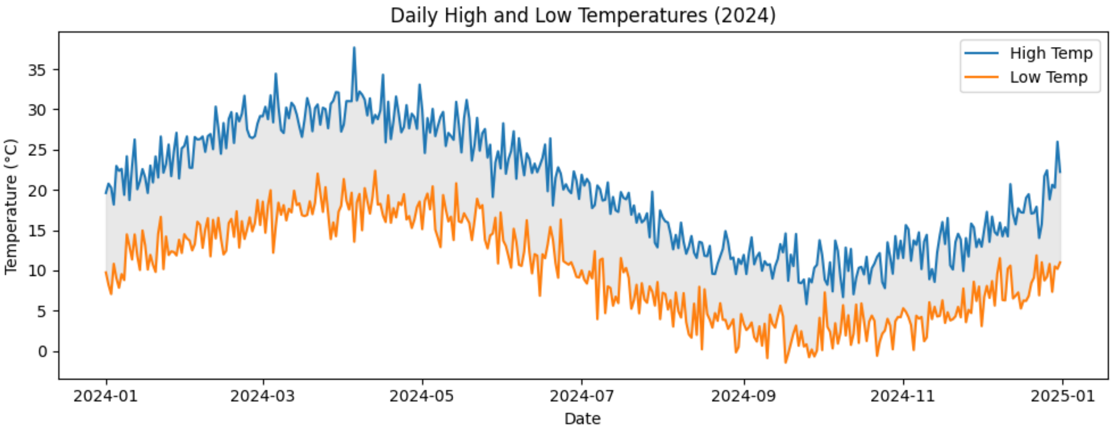




0.14 Real-World Data Examples

0.15 Weather Data Example (NOAA)

[26]:
```
import pandas as pd

# Simulate daily high and low temperatures for a year
dates = pd.date_range('2024-01-01', '2024-12-31')
day_of_year = np.arange(len(dates))
# Seasonal pattern plus some noise
high_temp = 20 + 10 * np.sin(2 * np.pi * day_of_year / 365) + np.random.
 ↪normal(0, 2, size=len(dates))
low_temp = 10 + 8 * np.sin(2 * np.pi * day_of_year / 365) + np.random.normal(0,␣
 ↪2, size=len(dates))
# Ensure low_temp is not above high_temp
low_temp = np.minimum(low_temp, high_temp - 1)

# Create a DataFrame to mimic reading from a CSV
weather = pd.DataFrame({'High': high_temp, 'Low': low_temp}, index=dates)

plt.figure(figsize=(10,4))
plt.plot(weather.index, weather['High'], label='High Temp')
plt.plot(weather.index, weather['Low'], label='Low Temp')
# Fill the area between high and low temperatures
plt.fill_between(weather.index, weather['Low'], weather['High'],␣
 ↪color='lightgray', alpha=0.5)

plt.title('Daily High and Low Temperatures (2024)')
plt.xlabel('Date')
plt.ylabel('Temperature (°C)')
plt.legend(loc='upper right')
plt.tight_layout()
plt.show()
```


Daily High and Low Temperatures (2024)
High Temp
Low Temp
Temperature (°C)
Date