Data Visualization in Python

In this chapter, we will explore how to visualize various data using Python with the renowned **Matplotlib** library. Matplotlib is one of the most important libraries dedicated to data visualization, providing powerful tools for creating graphs and charts. In addition to Matplotlib, there are several other libraries available for data plotting in Python, each with its unique features that make them suitable for specific scenarios. In this chapter, we will review some of these libraries and their key features.

A brief overview of data visualization libraries in Python:

1. **Matplotlib**
   - A foundational library supporting various chart types like line, bar, and histograms. Flexible and customizable (changeable figures).
2. **Seaborn**
   - Built on Matplotlib, adds support for statistical plots like box plots and distribution plots. Ideal for **statistical** data.
3. **Plotly**
   - An interactive library with **3D** and geographic charts, used for applications and interactive dashboards (**Interactive data display**).
4. **Bokeh**
   - **Create interactive charts for the web**, suitable for real-time data analysis and dashboards (Display analysis results directly without delay). For example: dashboard for monitoring the stock market.
5. **Altair**
   - A declarative library follows a clear and simple method of use, making it suitable for beginners and users who want to get quick results. Great for large and complex data visualizations.
6. **Pandas**
   - Enables quick data plotting directly from Pandas **DataFrames**, perfect for quick insights during data analysis.
7. **Geopandas**
   - Built on Pandas, it supports geographic plots like vector maps, useful for spatial analysis and geographic data.

You can install the Matplotlib library using the following command in the command prompt:

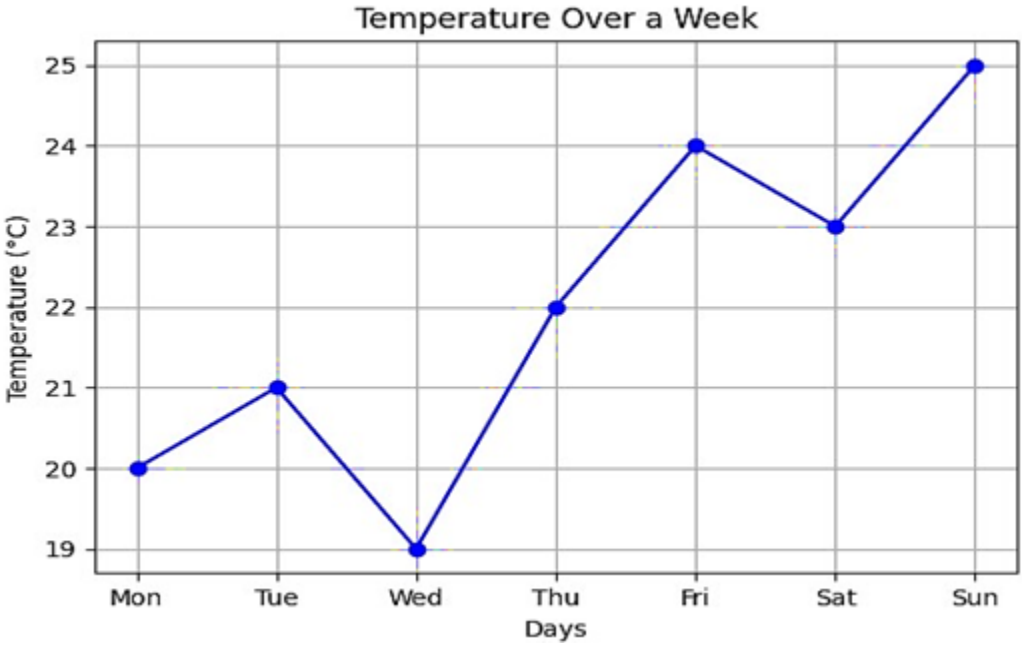
```
pip install matplotlib
```

3. Basic Types of Plots

❖ Line Plot

import matplotlib.pyplot as plt

```
# Atmospheric data
temperature = [20, 21, 19, 22, 24, 23, 25]
days = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat',
'Sun']
#plt.figure(figsize=(10, 5))# to change figure size
plt.plot(days, temperature, marker='o', linestyle='-',
color='b',linewidth=2)
plt.title('Temperature Over a Week')
plt.xlabel('Days')
plt.ylabel('Temperature (°C)')
#plt.text(0, 25, 'Max Temp', fontsize=12, color='red')
#plt.grid()
plt.show()
```



Develop your diagram:

- plt.figure(figsize=(10, 5))# to change figure size
- plt.grid()

• إضافة النصوص والتسميات

plt.text(0, 25, 'Max Temp', fontsize=12, color='red')# توضع اعلى ايعاز رسم الشبكة

سيضيف تعليقًا باسم "Max Temp" في موقع محدد عند النقطة (0, 25)، حيث يمثل 0 موضع اليوم على محور(x) ، و 25 هو ارتفاع النص على محور (y). يمثل mon الاحداثي صفر و تمثل sun الاحداثي 6 و بهذا يمكن التحكم في وضع النص داخل الشكل او فوق النقاط الصغرى و العظمى.

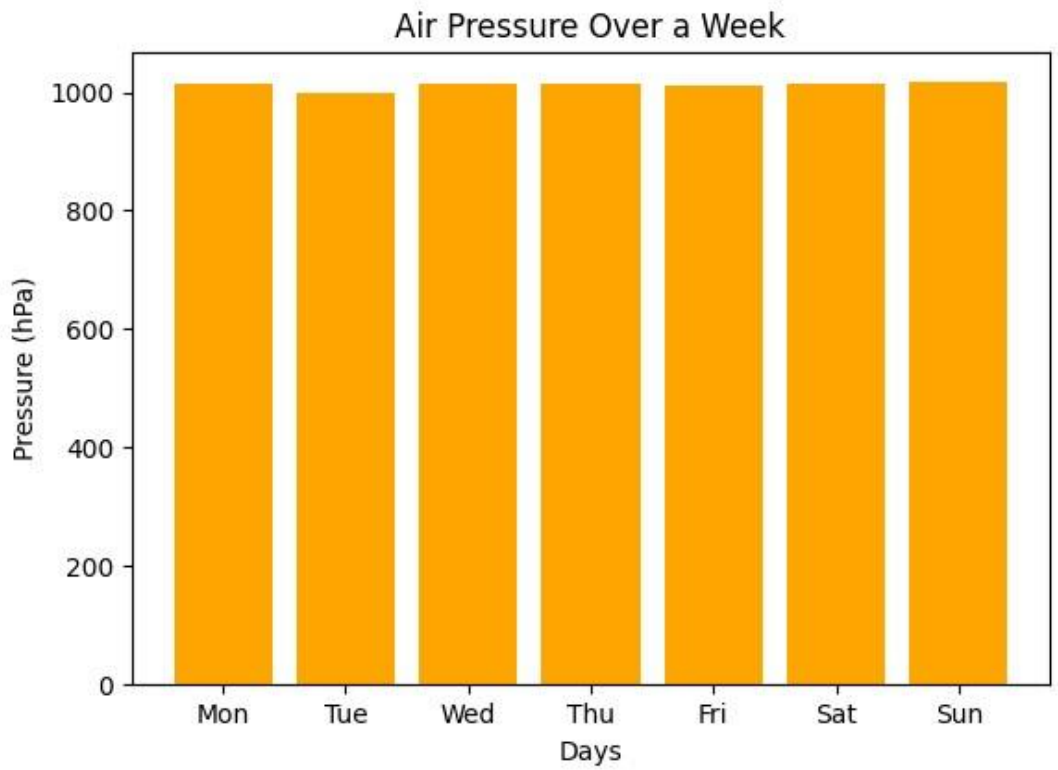
• تخصيص الخطوط والأحجام

plt.xlabel('Days', fontsize=14, fontweight='bold')# بدلا من plt.xlabel('Days')

❖ Bar Plot

import matplotlib.pyplot as plt

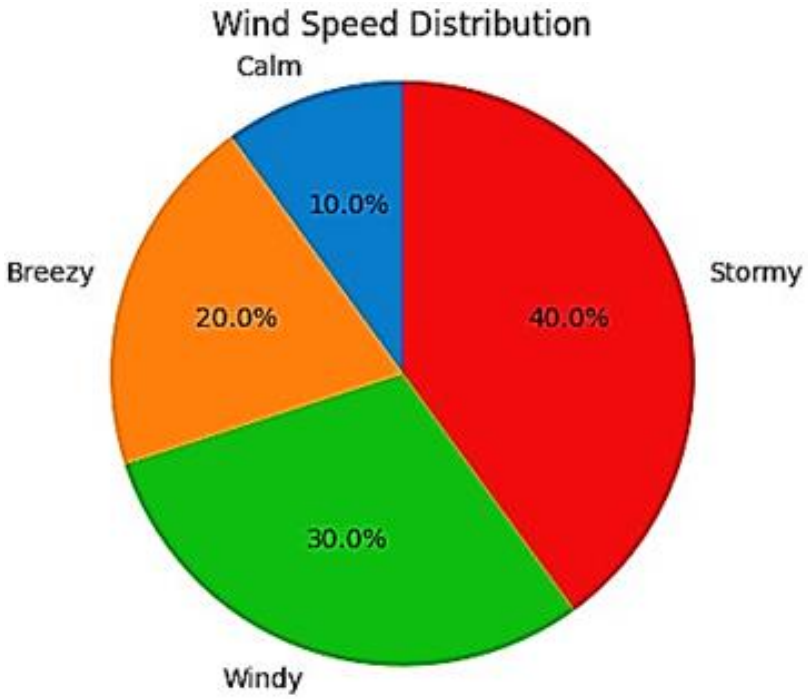
```
# Atmospheric data
pressure = [1012, 1010, 1015, 1013, 1011, 1012, 1014]
days = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat',
'Sun']

# رسم المخطط الشريطي
plt.bar(days, pressure, color='orange')
plt.title('Air Pressure Over a Week')
plt.xlabel('Days')
plt.ylabel('Pressure (hPa)')
plt.figure(figsize=(10, 5))# to change figure size
plt.show()
```



❖ Pie Chart

import matplotlib.pyplot as plt

```
wind_speed = [5, 10, 15, 20]
labels = ['Calm', 'Breezy', 'Windy', 'Stormy']

# رسم المخطط الدائري
plt.pie(wind_speed, labels=labels, autopct='%1.1f%%',
startangle=90)
plt.title('Wind Speed Distribution')
plt.axis('equal')
plt.show()
```



سيقوم البرنامج برسم wind_speed مع lables.

plt.pie :دالة ترسم المخطط الدائري.

wind_speed: القيم التي سيتم تمثيلها في المخطط.

labels=labels: يضيف التسميات المحددة لكل جزء من المخطط. هي وسيلة لربط كل قيمة سرعة رياح بالتسمية المناسبة لها مثلا: القيمة 5 من wind_speed ستُعطى التسمية "Calm".اي ان عنوان الشكل هو اسماء القائمة lable.

autopct='%1.1f%%': يضيف النسب المئوية لكل جزء داخل المخطط الدائري، حيث يتم عرض نسبة مئوية بمرتبة عشرية واحدة.

startangle=90: يبدأ الرسم من زاوية 90 درجة، مما يجعل أول جزء calm في الأعلى .اي سيبدا من 360 باعتباره 90.

plt.axis('equal'): يجعل أبعاد المخطط متساوية لإظهار المخطط بشكل دائرى.

%%في النهاية يعني أنه سيتم عرض علامة النسبة المئوية (%) في النص. إذا كنت تستخدم % فقط، فسيفسر كرمز تنسيق، لذا تحتاج إلى استخدام %% للإشارة إلى أنه يجب عرضها كعلامة النسبة المئوية.

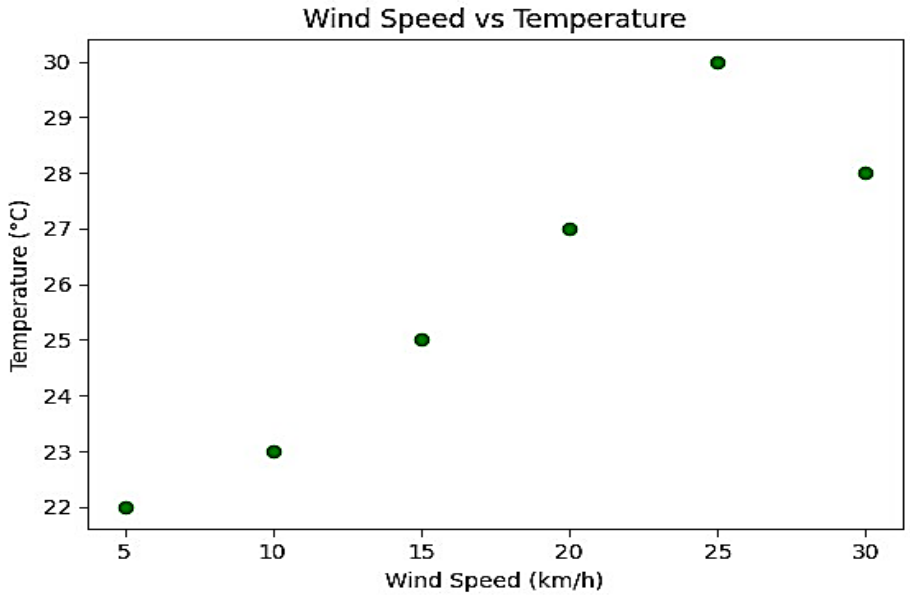
- إذا كانت النسبة المقابلة لقيمة 5 (Calm)هي 10%، فسيظهر هذا الجزء في المخطط على أنه "10.0%"
- إذا كانت النسبة المقابلة لقيمة 10 (Breezy) هي 20%، فسيظهر "20.0%" وهكذا لبقية الأجزاء.

- **وضوح البيانات** :يوفر autopct طريقة فعالة لجعل البيانات أكثر وضوحًا، مما يسهل على المشاهدين فهم النسب المئوية لكل فئة بسرعة.

❖ Scatter Plot

import matplotlib.pyplot as plt

```
# wind and ambient temperature data
wind_speeds = [5, 10, 15, 20, 25, 30]
temperature = [22, 23, 25, 27, 30, 28]

# رسم المخطط النقطي
plt.scatter(wind_speeds, temperature, color='green')
plt.title('Wind Speed vs Temperature')
plt.xlabel('Wind Speed (km/h)')
plt.ylabel('Temperature (°C)')
plt.show()
```



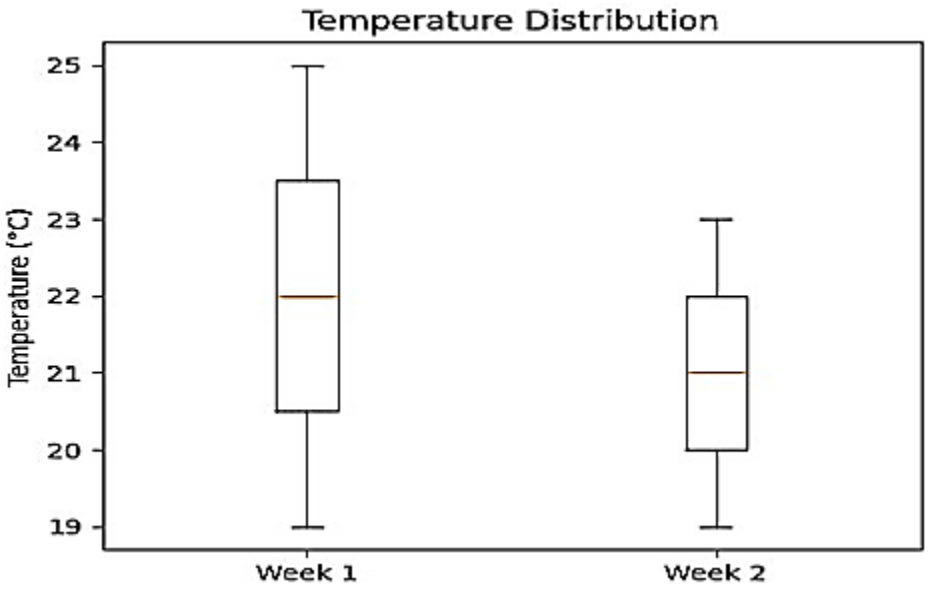
❖ Box Plot

import matplotlib.pyplot as plt

```
temperature = [22, 23, 25, 21, 20, 19]
data = [temperature, [20, 22, 19, 21, 20, 23, 22]]

plt.boxplot(data, labels=['Week 1', 'Week 2'])
plt.title('Temperature Distribution')
plt.ylabel('Temperature (°C)')
```

Week 1 : [22, 23, 25, 21, 20, 19].
Week 2: [20, 22, 19, 21, 20, 23, 22].

```
plt.show()
```

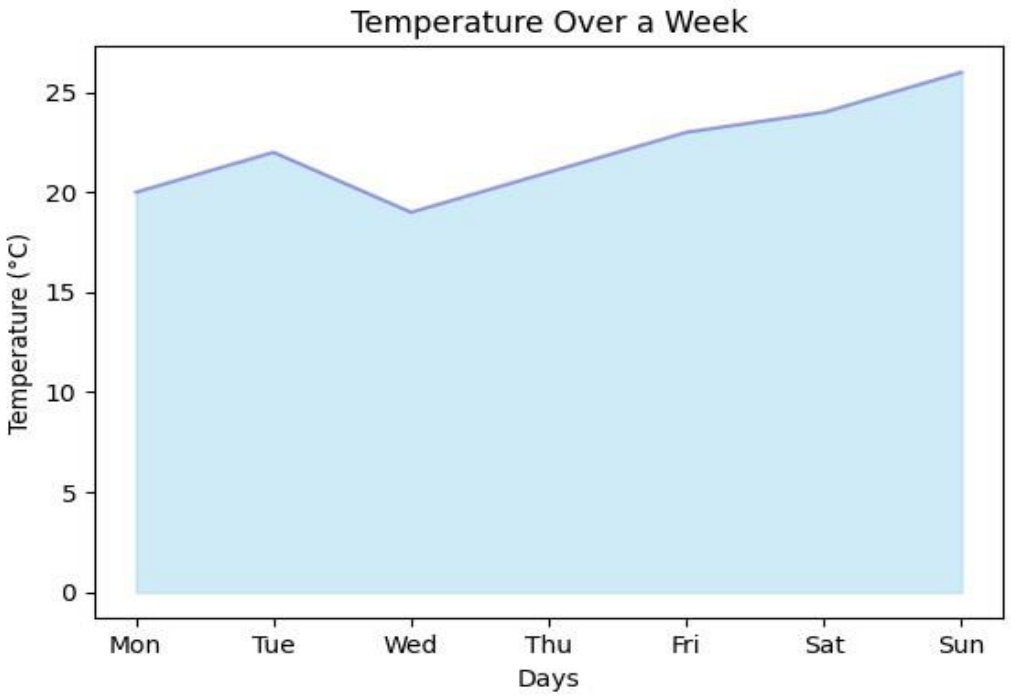


❖ Area Plot

```
import matplotlib.pyplot as plt

#Atmospheric Data
days = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat',
'Sun']
temperature = [20, 21, 19, 22, 24, 23, 25]

# رسم المخطط السطحي
plt.fill_between(days, temperature, color='skyblue',
alpha=0.4)
plt.plot(days, temperature, color='Slateblue',
alpha=0.6)
plt.title('Temperature Over a Week')
plt.xlabel('Days')
plt.ylabel('Temperature (°C)')
plt.show()
```

fill_between: تملأ المنطقة الواقعة بين محور الأيام (days) وقيم درجة الحرارة (temperature) بلون معين، مما يخلق مخططًا سطحيًا.

- **color='skyblue'** : يحدد لون التظليل (اللون السماوي).
- **alpha=0.4**: يحدد شفافية اللون، حيث أن القيمة 0.4 تجعل اللون شبه شفاف.
- **plt.plot**: يرسم خطًا يمثل البيانات (درجات الحرارة عبر الأيام).
- **color='Slateblue'**: يحدد لون الخط (اللون الأزرق الأردوازي).
- **alpha=0.6** : يحدد شفافية الخط، بحيث يكون أقل شفافية من التظليل تحت الخط.



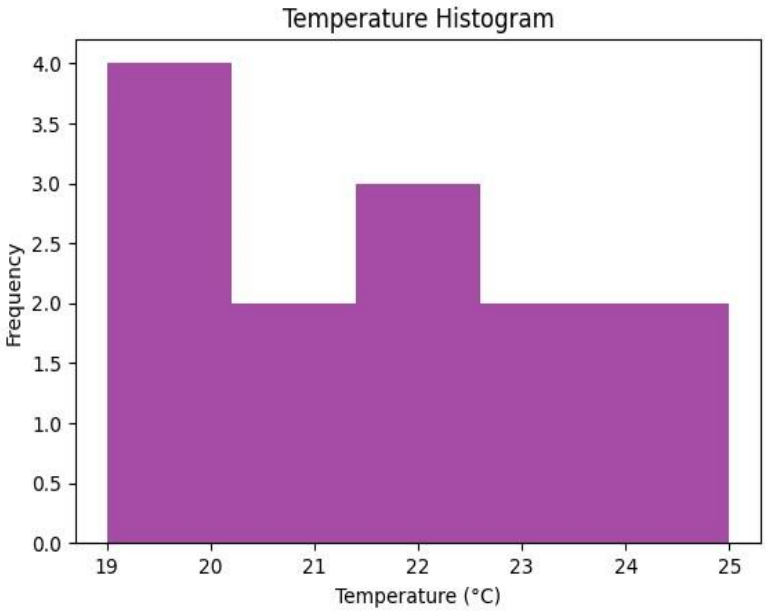
❖ Histogram

```
import matplotlib.pyplot as plt

temp_data = [20, 21, 19, 22, 24, 23, 25, 20, 22, 21,
23, 19, 22]

# Plot histogram diagram
plt.hist(temp_data, bins=5, color='purple', alpha=0.7)
plt.title('Temperature Histogram')
plt.xlabel('Temperature (°C)')
plt.ylabel('Frequency')
plt.show()
```

1. **تحديد نطاق البيانات (Data Range):** في البداية، يتم تحديد النطاق الكلي للبيانات، وهو الفرق بين القيمة الأصغر والأكبر في القائمة.
   - على سبيل المثال، إذا كانت القيم `[20, 21, 19, 22, 24, 23, 25, 20, 22, 21, 23, 19, 22]`، فإن النطاق هو من 19 إلى 25.
2. **تقسيم النطاق إلى عدد من الحاويات (bins):** عند تحديد `bins=5`، فإن النطاق سيتم تقسيمه إلى 5 حاويات متساوية.
   - عرض الحاوية الواحدة يحسب كالتالي:

$$\text{عرض الحاوية} = \frac{\text{النطاق}}{\text{عدد الحاويات}} = \frac{19 - 25}{5} = 1.2$$

   - بالتالي، ستكون الحاويات كالتالي:
   - الحاوية 1: من 19 إلى 20.2
   - الحاوية 2: من 20.2 إلى 21.4
   - الحاوية 3: من 21.4 إلى 22.6
   - الحاوية 4: من 22.6 إلى 23.8
   - الحاوية 5: من 23.8 إلى 25



حساب التكرارات لكل حاوية:

- **الحاوية الأولى (19 - 20.2)**: تحتوي على القيم 19, 20, 20, 19، أي 4 تكرارات.
- **الحاوية الثانية (20.2 - 21.4)**: تحتوي على القيم 21, 21، أي 2 تكرار.
- **الحاوية الثالثة (21.4 - 22.6)**: تحتوي على القيم 22, 22, 22، أي 3 تكرارات.
- **الحاوية الرابعة (22.6 - 23.8)**: تحتوي على القيم 23, 23، أي 2 تكرار.
- **الحاوية الخامسة (23.8 - 25)**: تحتوي على القيم 24, 25، أي 2 تكرار.

في شكل المضلع التكراري هناك ثلاث تكرارات متساوية لذا ظهرت قمتين فقط .

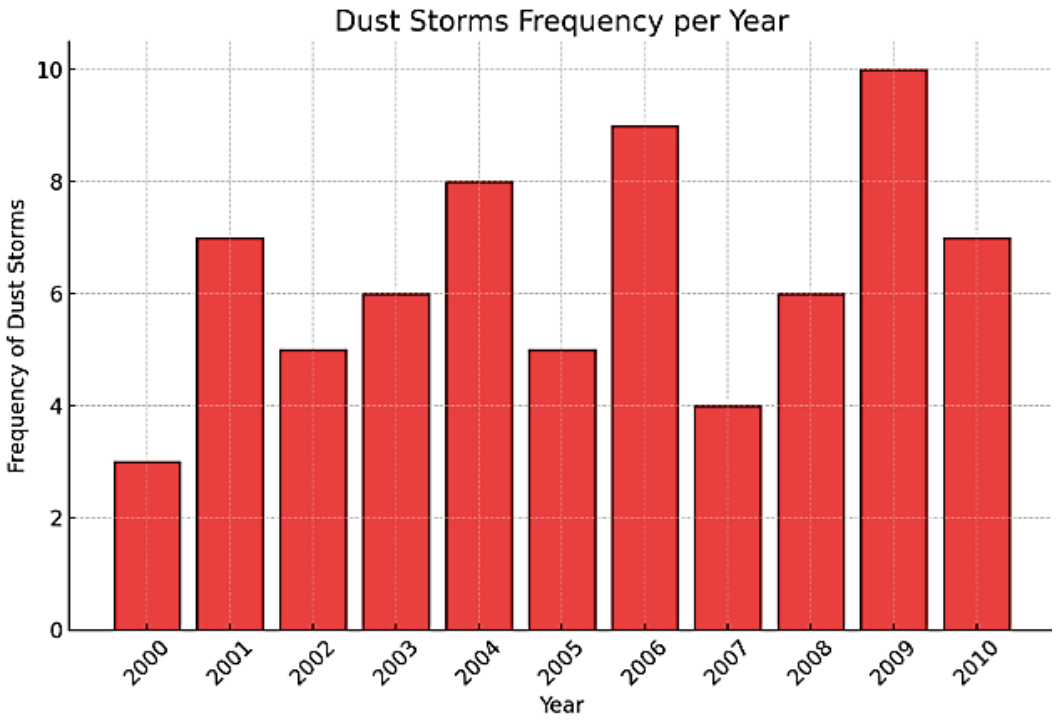
Example 2

```
import matplotlib.pyplot as plt

years = [2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008,
2009, 2010]

dust_storms = [3, 7, 5, 6, 8, 5, 9, 4, 6, 10, 7] #  لـكل سنة عدد
الـعواصف

# plot bars

plt.bar(years, dust_storms, color='brown', alpha=0.7,
edgecolor='black')

plt.title('Dust Storms Frequency per Year')

plt.xlabel('Year')

plt.ylabel('Frequency of Dust Storms')

plt.xticks(years, rotation=45)  #

plt.grid(axis='y', linestyle='--', alpha=0.7)  #

plt.show()
```

plt.xticks: هي دالة تُستخدم لتعيين القيم والعناوين لمحور x.

rotation=45 : يقوم بتدوير النصوص الخاصة بالسنوات (اسماء السنين) بزاوية 45 درجة، بحيث تظهر العناوين بشكل مائل. هذا يكون مفيدًا عندما تكون العناوين طويلة أو متقاربة، مما يجعلها أكثر وضوحًا ويسهل قراءتها.

'axis='y : يحدد أن الشبكة ستظهر فقط على طول المحور y (الرأسي).

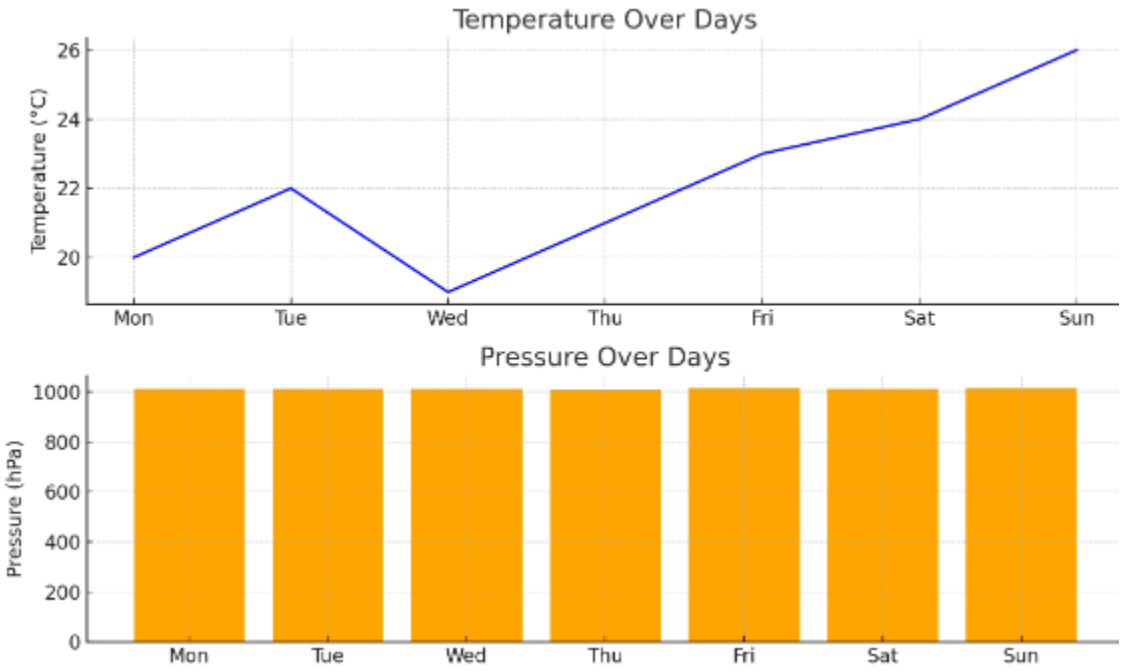


Subplots

The subplot function in Matplotlib allows splitting the plotting area into a grid of subplots within a single figure, making it easier to display multiple charts side by side to **compare** and show different relationships between variables.

Example 1:

```
import matplotlib.pyplot as plt

days = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat',
'Sun']
temperature = [20, 22, 19, 21, 23, 24, 26]
pressure = [1012, 1011, 1013, 1010, 1014, 1011, 1015]

fig, axs = plt.subplots(2, 1)  # صفين
axs[0].plot(days, temperature, color='blue')
axs[1].bar(days, pressure, color='orange')
plt.show()
```



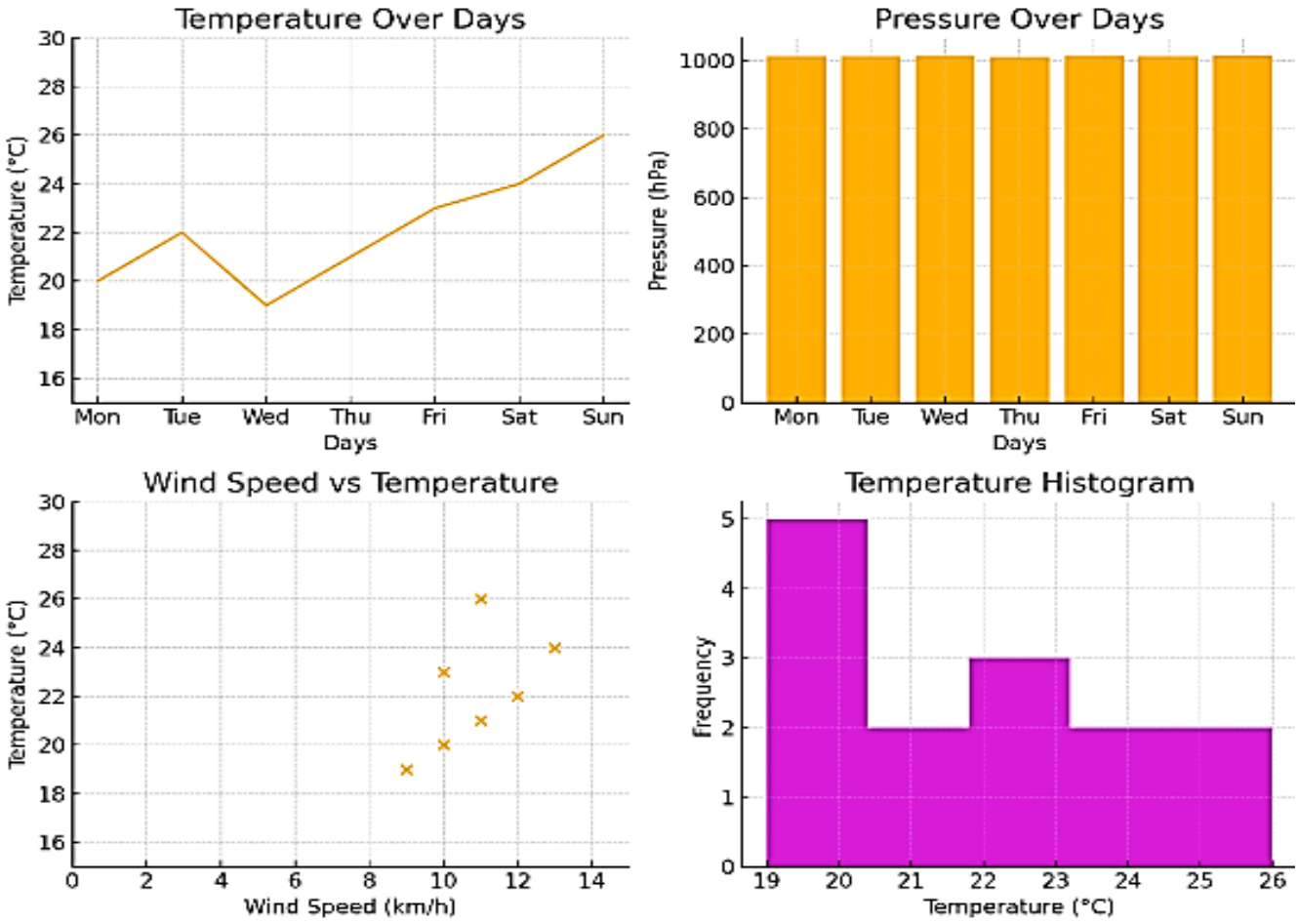
Example 2:

```
import matplotlib.pyplot as plt

# Define the data for days, temperature, pressure, wind
speed, and temperature histogram

days = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat',
'Sun']

temperature = [20, 22, 19, 21, 23, 24, 26]

pressure = [1012, 1011, 1013, 1010, 1014, 1011, 1015]

wind_speeds = [10, 12, 9, 11, 10, 13, 11]

temp_data = [20, 22, 21, 19, 25, 24, 23, 22, 20, 19,
21, 24, 26, 20]

#  إنشاء الشكل والشبكة الفرعية 2*2

fig, axs = plt.subplots(2, 2, figsize=(10, 8))

# رسم البيانات على المخططات الفرعية
```

```
axs[0, 0].plot(days, temperature)# first diagram

axs[0, 0].set_title('Temperature Over Days')

axs[0, 0].set_xlabel('Days')

axs[0, 0].set_ylabel('Temperature (°C)')

axs[0, 0].set_ylim(15, 30)  # تحديد مدى المحور الصادي
فقط لان المحور السيني ايام لا يمكن تغيير مداها كبعدافقي.

axs[0, 1].bar(days, pressure)# second diagram

axs[0, 1].set_title('Pressure Over Days')

axs[0, 1].set_xlabel('Days')

axs[0, 1].set_ylabel('Pressure (hPa)')


axs[1, 0].scatter(wind_speeds, temperature)# third
diagram

axs[1, 0].set_title('Wind Speed vs Temperature')

axs[1, 0].set_xlabel('Wind Speed (km/h)')

axs[1, 0].set_ylabel('Temperature (°C)')

axs[1, 0].set_xlim(0, 7)  # تحديد مدى المحور السيني

axs[1, 0].set_ylim(15, 30)  # تحديد مدى المحور الصادي
(التحكم في مدى المقياس)

axs[1, 1].hist(temp_data, bins=5, color='purple',
alpha=0.7)# fourth diagram

axs[1, 1].set_title('Temperature Histogram')

axs[1, 1].set_xlabel('Temperature (°C)')

axs[1, 1].set_ylabel('Frequency')
```

```
#(ضبط التخطيط (لكي يظهر الشكل كله دون اقتطاع
plt.tight_layout()
plt.show()
```

axs[0, 0]	axs[0, 1]
axs[1, 0]	axs[1, 1]



Save and export

The resulting plots can be saved in various formats (PNG, PDF, SVG, etc.)

• Save the graphs as images.

```
plt.savefig('temperature_plot.png', dpi=300)
```

(SVG: Scalable Vector Graphics) هي صيغة ملفات تستخدم لعرض الرسومات ثنائية الأبعاد، مثل الأيقونات والرسوم البيانية، بجودة عالية وقابلة للتوسع على مختلف أحجام الشاشات دون فقدان الدقة. تعتمد SVG على **لغة XML** لتمثيل الرسومات، مما يجعلها نصية بالكامل وقابلة للتحرير، ويمكن تضمينها مباشرةً في صفحات الويب. تُعد هذه الصيغة شائعة لتصميمات الويب لأنها تدعم التفاعل، والحركة، وتعديلات الألوان بسهولة.

- **Different** file formats (PDF, SVG)

plt.savefig('temperature_plot.pdf')

Example:

```
import matplotlib.pyplot as plt

days = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat',
'Sun']
temperature = [20, 22, 19, 21, 23, 24, 26]
pressure = [1012, 1011, 1013, 1010, 1014, 1011, 1015]

fig, axs = plt.subplots(2, 1)  # two rows x on column
axs[0].plot(days, temperature, color='blue')
axs[1].bar(days, pressure, color='orange')
plt.savefig('weather_plot.png', dpi=300)
#plt.savefig('weather_plot.pdf')
plt.show()
```

إذا لم يتم تحديد مسار: إذا قمت بكتابة هذا السطر كما هو، دون تحديد مسار، فسيتم حفظ الملف في **المجلد الحالي** الذي تعمل فيه و هو المجلد الذي يحتوي على السكربت أو الملف الذي تنفذه .

To save figures in some location:

plt.savefig('V:/بايثون/المحاضرات/final lectures/temperature_plot.png', dpi=300)# عندما اريد خزنه في مجلد معين في حاسوبي الذي يعمل على نظام وزدوز

To save figure on Desktop:

Windows system: C:/Users/اسم المستخدم/Desktop/

macOS: /Users/اسم المستخدم/Desktop/

Linux: /home/اسم المستخدم/Desktop/

Plt.savefig('/home/alaa/Desktop/temperature.pdf')

Plt.savefig('/home/alaa/Documents/temperature.pdf')

يمكنك تغيير دقة الصورة عند حفظ الرسم باستخدام `plt.savefig()` عن طريق تعديل قيمة المعامل `dpi`. الرقم المحدد لـ `dpi` (نقاط لكل بوصة) يؤثر على وضوح الصورة وحجمها.

خيارات دقة الصورة:

- **72 dpi**: دقة منخفضة، مناسبة للعرض على الشاشات ولكن ليست مناسبة للطباعة.
- **150 dpi**: دقة متوسطة، تُستخدم عادةً للمستندات والمواد المطبوعة غير الرسمية.
- **300 dpi** : دقة عالية، مثالية للطباعة وضرورية لجودة عالية في الصور المطبوعة.
- **600 dpi أو أكثر**: دقة عالية جدًا، تستخدم في الحالات التي تتطلب تفاصيل دقيقة جدًا.

DPI تعني **"Dots Per Inch"**، وهي وحدة قياس تستخدم لتحديد دقة الصورة أو الطباعة. تعبر DPI عن عدد النقاط (أو البكسلات) التي يمكن وضعها في بوصة واحدة من الصورة.