

```
from IPython.display import Image, display
display(Image(filename='p1.png'))
```



```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
```

Descriptive analysis

Descriptive analysis is the process of summarizing and understanding the basic features of data in a DataFrame. It provides insights such as central tendency, distribution, and variability of data. Below is a step-by-step guide for performing descriptive analysis on a DataFrame using Python's pandas library.

Data visualization

Data visualization is a powerful way to represent information visually, making it easier to analyze, understand, and communicate insights. Depending on the type of data and the story you want to tell, different types of visualizations are more appropriate.

Types of Visualizations

- Multivariate Heatmap, Pairplot
Example: Correlation between features
- Categorical

Bar chart, Pie chart

Example: Gender ratio

- Numerical Histogram, Boxplot

Example: Age distribution

- Relationships

Scatter plot, Line plot

Example: Temp vs.Humidity

Bar Chart

A bar chart (or bar graph) is a graphical representation of data using rectangular bars. Each bar's length (or height) is proportional to the value it represents. Bar charts are commonly used to compare different categories or groups.

Axes:

The x-axis typically represents the categories being compared.
The y-axis represents the numerical values or frequencies.

1. Basic Vertical Bar Chart

Displays individual category values using vertical bars.

Each bar's height represents its value (e.g., sales, counts, etc.).

This is the most common type of bar chart.

Use When:

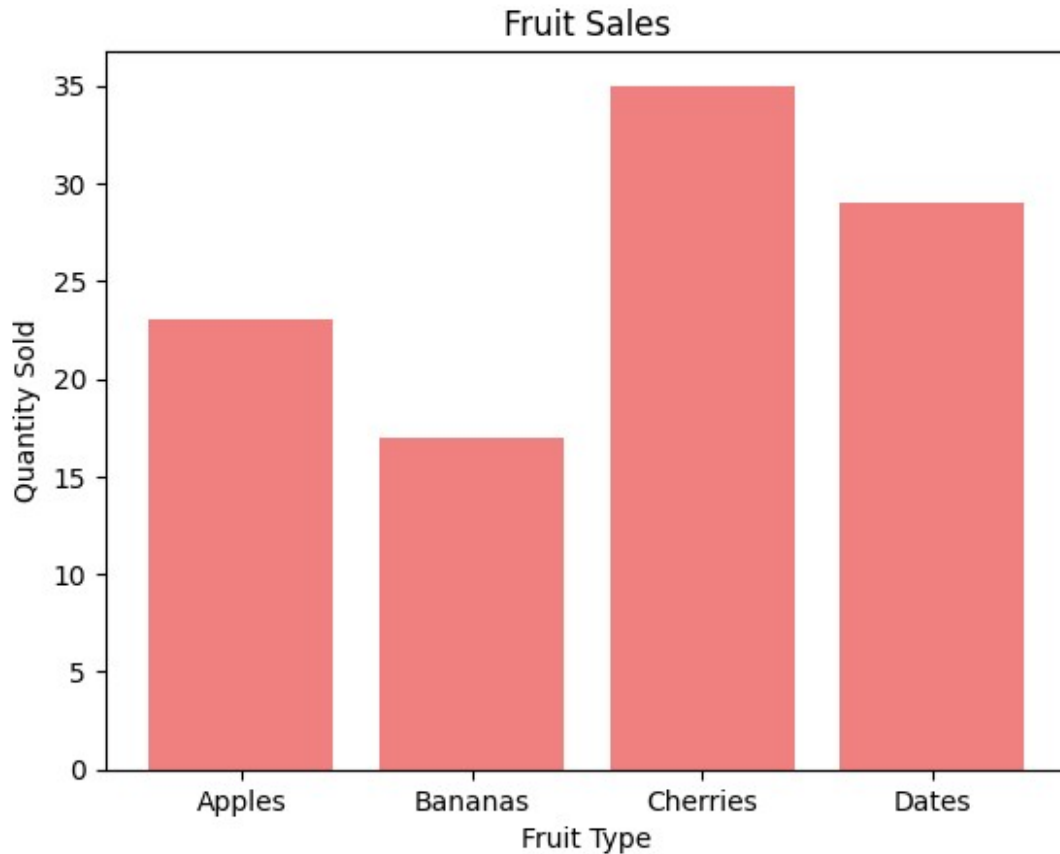
You want to compare quantities across different categories.

Example

comparing sales of different fruit types.

```
fruits = ['Apples', 'Bananas', 'Cherries', 'Dates']
sales = [23, 17, 35, 29]

plt.bar(fruits, sales, color='lightcoral')
plt.title('Fruit Sales')
plt.xlabel('Fruit Type')
plt.ylabel('Quantity Sold')
plt.show()
```



2. Horizontal Bar Chart

Similar to a vertical bar chart, but bars are horizontal.

The length of each bar represents its value.

Use When:

Category names are long and fit better on the y-axis.

You want to emphasize rank or order (e.g., top countries by GDP).

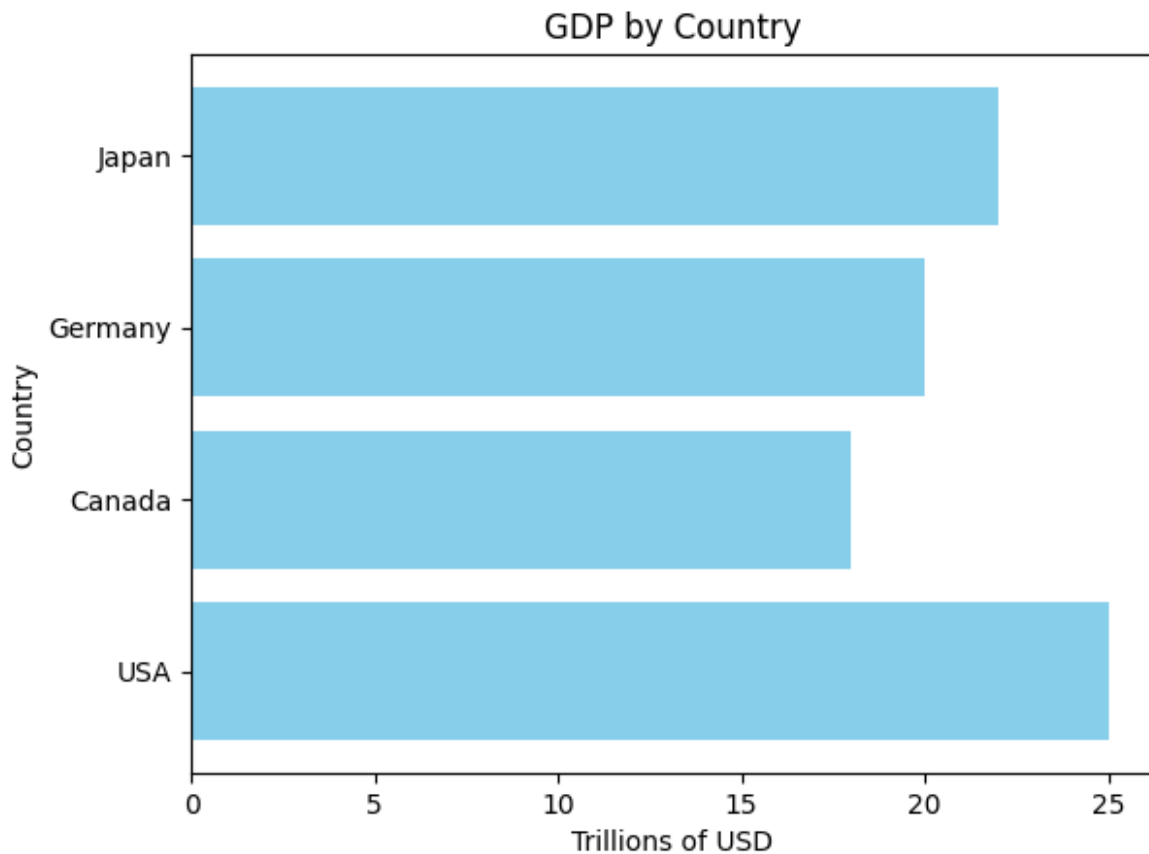
Example Output:

Bars extend sideways along the x-axis.

```
countries = ['USA', 'Canada', 'Germany', 'Japan']
gdp = [25, 18, 20, 22]

plt.barh(countries, gdp, color='skyblue')
plt.title('GDP by Country')
plt.xlabel('Trillions of USD')
```

```
plt.ylabel('Country')  
plt.show()
```



3. Grouped Bar Chart (Comparing Two Categories)

Displays multiple bars per category, grouped side by side.

Used to compare multiple datasets across the same categories.

Use When:

You want to compare two or more items across several categories.

Example:

comparing Product A vs Product B sales each year.

Each group (like "2021") has two bars next to each other.

```
import numpy as np  
  
years = ['2021', '2022', '2023']  
sales_A = [25, 30, 35]
```

```

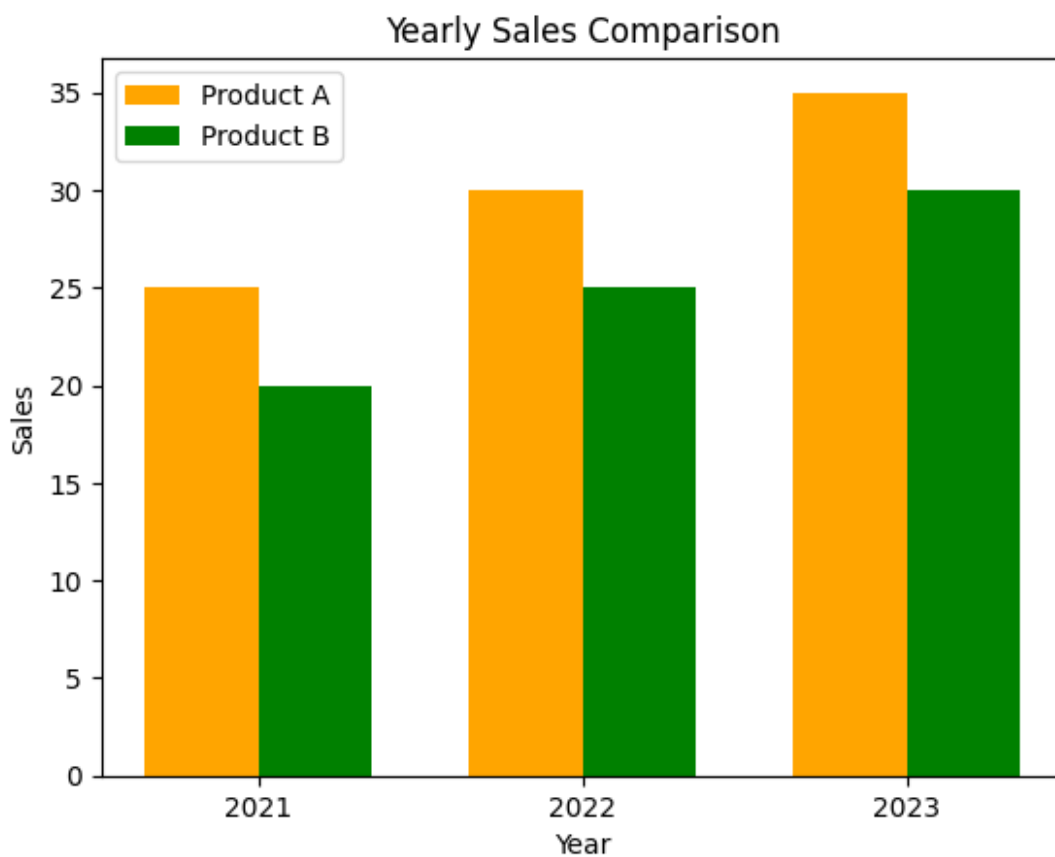
sales_B = [20, 25, 30]

x = np.arange(len(years))
width = 0.35

plt.bar(x - width/2, sales_A, width, label='Product A',
color='orange')
plt.bar(x + width/2, sales_B, width, label='Product B', color='green')

plt.title('Yearly Sales Comparison')
plt.xlabel('Year')
plt.ylabel('Sales')
plt.xticks(x, years)
plt.legend()
plt.show()

```



4. Stacked Bar Chart

Shows subcategories stacked on top of each other within a single bar.

The total height represents the overall value (sum of components).

Use When:

You want to show part-to-whole relationships and overall totals.

Example:

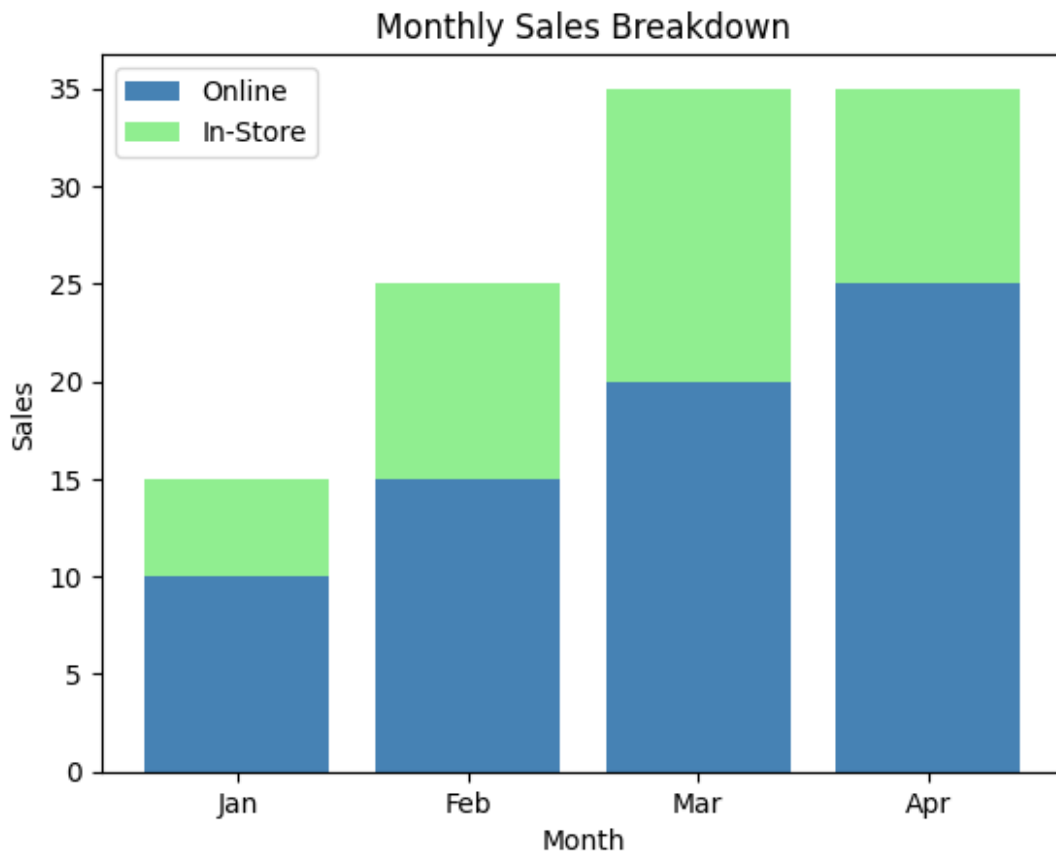
Online vs In-store sales for each month.

Each bar shows stacked sections, making totals and proportions visible.

```
months = ['Jan', 'Feb', 'Mar', 'Apr']
online_sales = [10, 15, 20, 25]
store_sales = [5, 10, 15, 10]

plt.bar(months, online_sales, color='steelblue', label='Online')
plt.bar(months, store_sales, bottom=online_sales, color='lightgreen',
label='In-Store')

plt.title('Monthly Sales Breakdown')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.legend()
plt.show()
```



Pie Chart

1. Basic Pie Chart

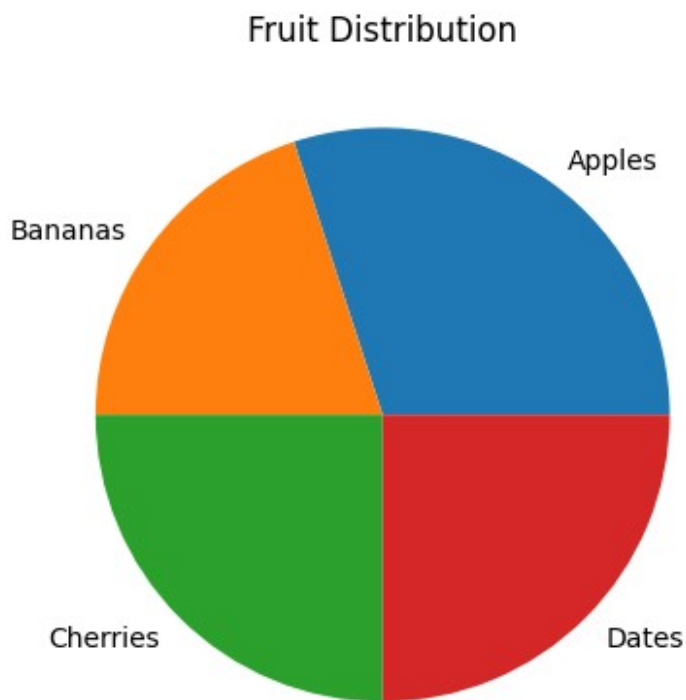
The simplest pie chart — shows how parts (fruits) contribute to a total.

Each slice's size represents its proportion.

Use When:

You want to show percentages or proportions of a whole

```
labels = ['Apples', 'Bananas', 'Cherries', 'Dates']  
sizes = [30, 20, 25, 25]  
  
plt.pie(sizes, labels=labels)  
plt.title('Fruit Distribution')  
plt.show()
```



2. Pie Chart with Percentages

Adds percentage labels using autopct.

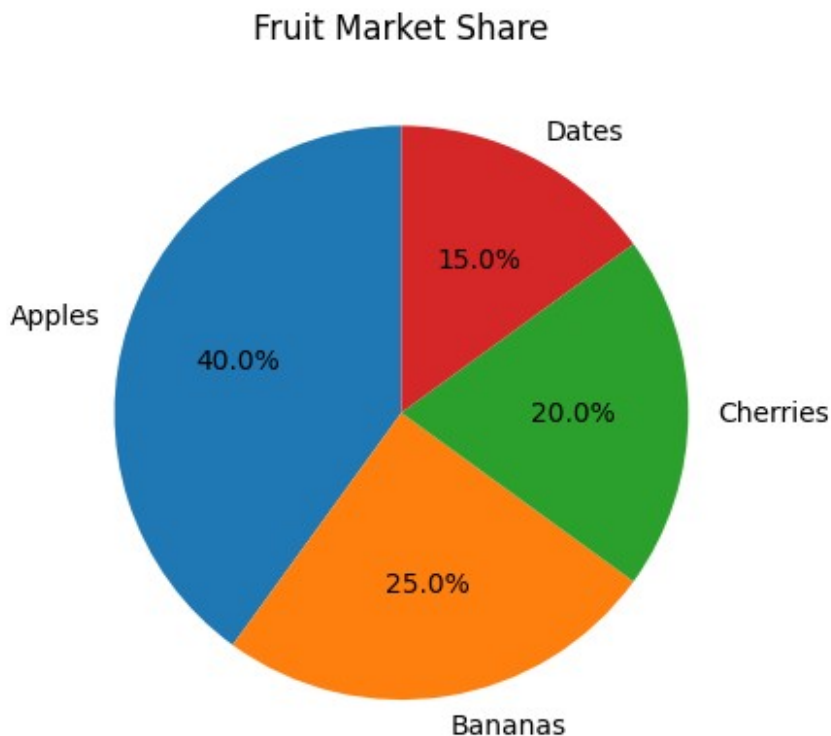
startangle=90 rotates the first slice to start from the top.

Use When:

You want to display exact percentages on the chart.

```
labels = ['Apples', 'Bananas', 'Cherries', 'Dates']
sizes = [40, 25, 20, 15]

plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
plt.title('Fruit Market Share')
plt.show()
```



3. Exploded (Highlighted) Pie Chart

“Explodes” one slice to highlight it (Apples).

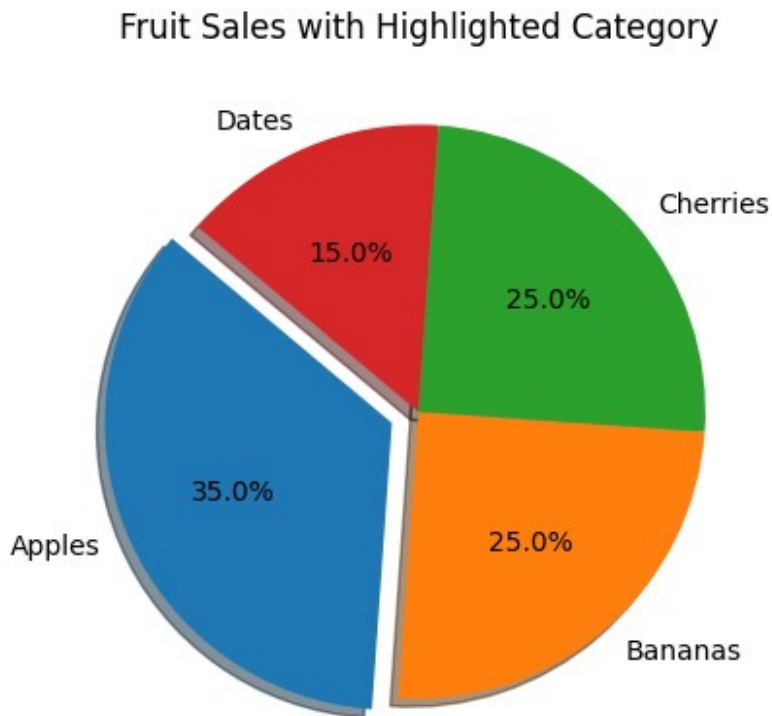
shadow=True adds a 3D-like shadow effect.

Use When:

You want to emphasize a specific category.

```
labels = ['Apples', 'Bananas', 'Cherries', 'Dates']
sizes = [35, 25, 25, 15]
explode = [0.1, 0, 0, 0] # “explode” the first slice
```

```
plt.pie(sizes, labels=labels, autopct='%1.1f%%', explode=explode,
shadow=True, startangle=140)
plt.title('Fruit Sales with Highlighted Category')
plt.show()
```



4. Ring Chart

A pie chart with a hole in the middle.

Easier to read and often used in dashboards.

Use When:

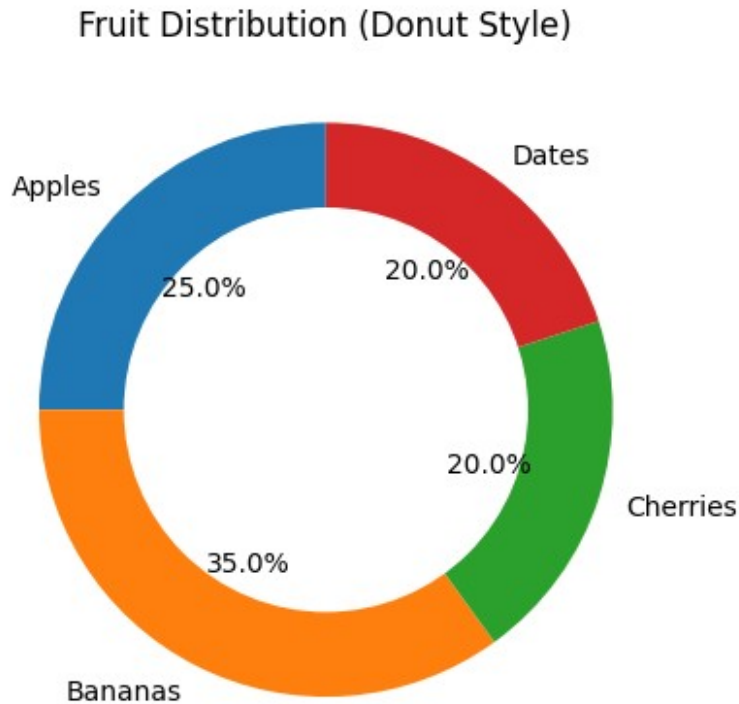
You want a modern, clean look for showing proportions.

```
labels = ['Apples', 'Bananas', 'Cherries', 'Dates']
sizes = [25, 35, 20, 20]

wedges, texts, autotexts = plt.pie(sizes, labels=labels,
autopct='%1.1f%%', startangle=90)
# Add a white circle to create a "donut" effect
centre_circle = plt.Circle((0,0), 0.70, color='white', fc='white')
fig = plt.gcf()
```

```
fig.gca().add_artist(centre_circle)

plt.title('Fruit Distribution (Donut Style)')
plt.show()
```



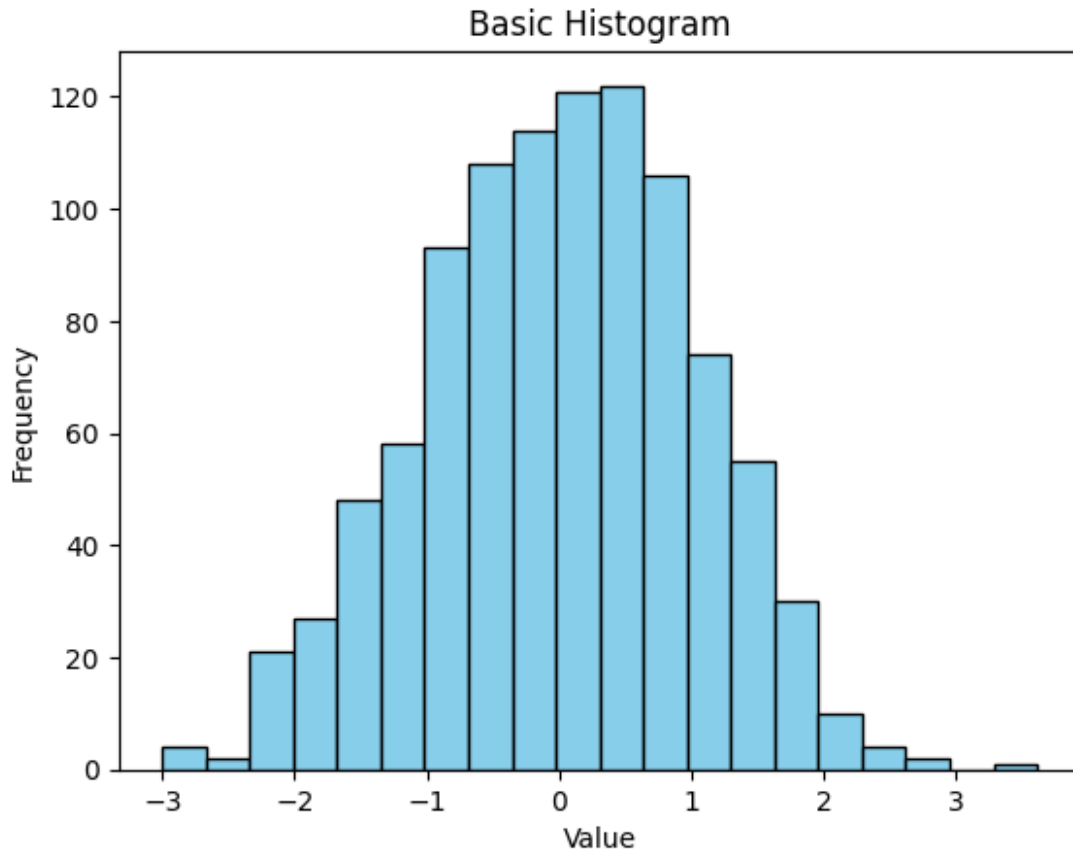
Histogram

A histogram shows how data is distributed — useful for seeing the frequency of values or the shape of your data (e.g., normal, skewed, bimodal).

Description: Shows how often data points fall within each range (bin). Useful for understanding spread and center of continuous data.

```
data = np.random.randn(1000)

plt.hist(data, bins=20, color='skyblue', edgecolor='black')
plt.title('Basic Histogram')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```



This image shows a basic histogram representing the distribution of a continuous variable labeled "Value."

Here's a breakdown of what it depicts:

X-axis (Value): Represents the range of values of the data. The values approximately span from -3 to +3.

Y-axis (Frequency): Represents how often each range (or "bin") of values occurs in the dataset.

Bars: Each bar's height shows the number of data points (frequency) that fall within a particular range of values.

Shape: The histogram shows a bell-shaped (normal) distribution, meaning most of the values are centered around 0, with frequencies tapering off symmetrically on both sides.

Color and style: The bars are filled with a light blue color (alpha=0.7 for transparency) and outlined in black, giving a clear visual distinction between bins.

Interpretation: This histogram likely represents a dataset sampled from a normal (Gaussian) distribution with a mean near 0 and a standard deviation around 1. Most observations are concentrated around 0, and very few appear in the tails beyond ± 2 or ± 3 .

Boxplot

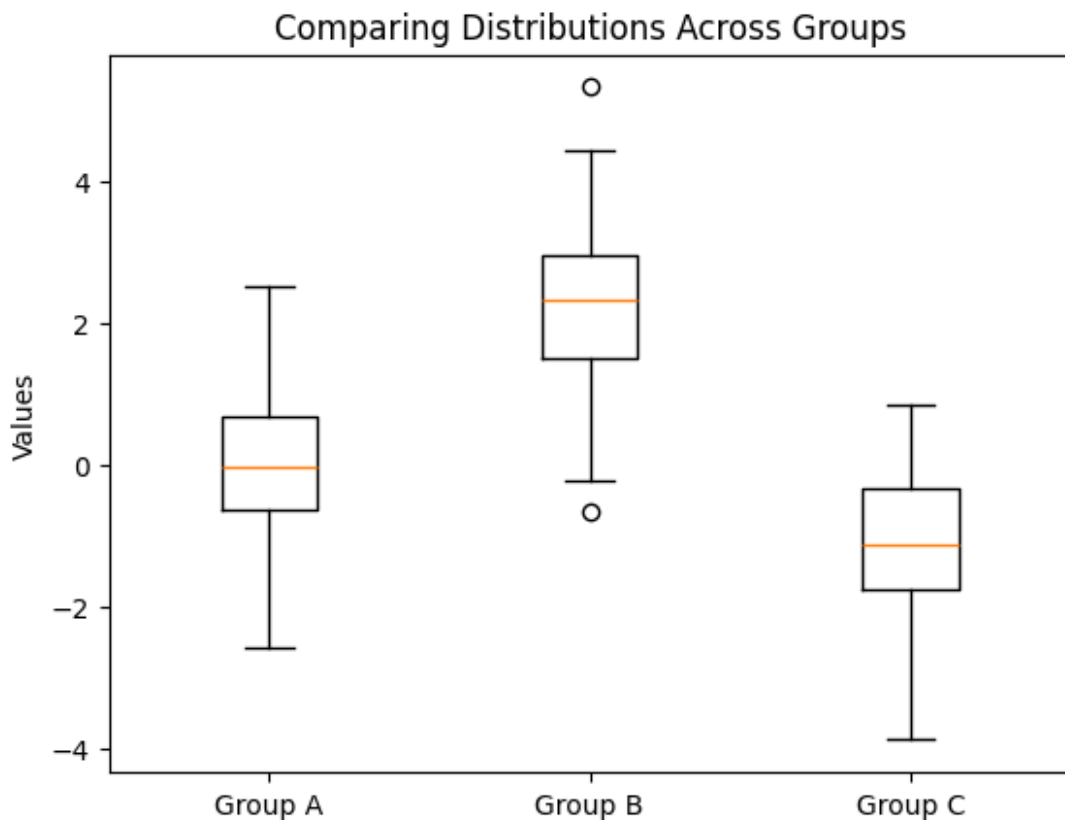
A boxplot shows median, quartiles, and outliers — great for comparing data spread or detecting anomalies. Displays the median (center line), IQR (box), and outliers (points) for a single dataset

Example 2: Multiple Boxplots (Comparison)

```
data = [np.random.randn(100), np.random.randn(100) + 2,  
np.random.randn(100) - 1]  
  
plt.boxplot(data, labels=['Group A', 'Group B', 'Group C'])  
plt.title('Comparing Distributions Across Groups')  
plt.ylabel('Values')  
plt.show()
```

C:\Users\Compumarts\AppData\Local\Temp\ipykernel_7532\3836022786.py:3:
MatplotlibDeprecationWarning: The 'labels' parameter of boxplot() has
been renamed 'tick_labels' since Matplotlib 3.9; support for the old
name will be dropped in 3.11.

```
plt.boxplot(data, labels=['Group A', 'Group B', 'Group C'])
```



Scatter plot

A scatter plot is a type of graph used to show the relationship between two numerical (quantitative) variables.

A scatter plot is a type of graph used to display the relationship or correlation between two numerical variables. It consists of points (dots), where each point represents an observation in the dataset. The position of each point is determined by the values of two variables: one plotted along the x-axis and the other along the y-axis.

Each dot on the graph shows the values of both variables for one observation.

Purpose

Scatter plots are mainly used to:

- Visualize relationships or correlations between variables (positive, negative, or none)

Positive correlation → as X increases, Y increases

Negative correlation → as X increases, Y decreases

No correlation → no clear pattern between X and Y

- Identify patterns or trends in data
- Detect outliers (points that don't fit the general pattern)

Example

If we plot students' study hours (X) vs. exam scores (Y):

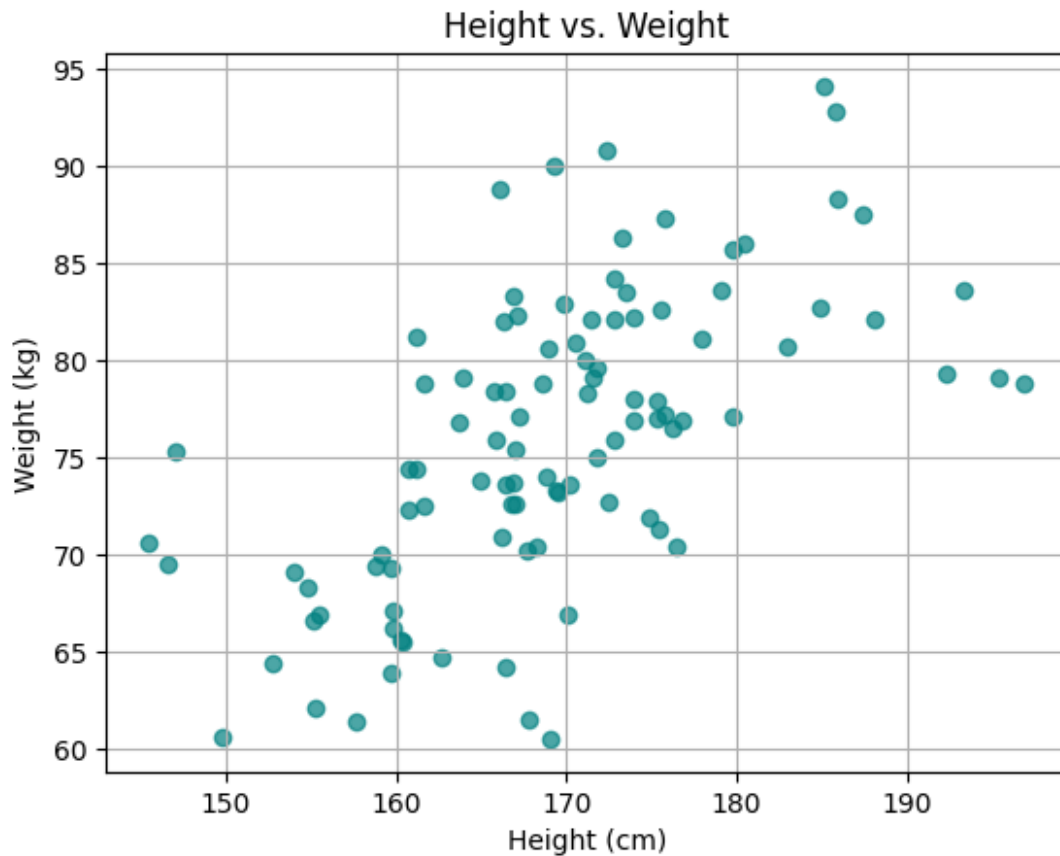
Each point = one student's data

If the dots go upward (as study hours increase, scores increase), it shows a positive correlation.

□ Example 1: Height vs. Weight

```
# Generate random data
height = np.random.normal(170, 10, 100)
weight = height * 0.45 + np.random.normal(0, 5, 100)

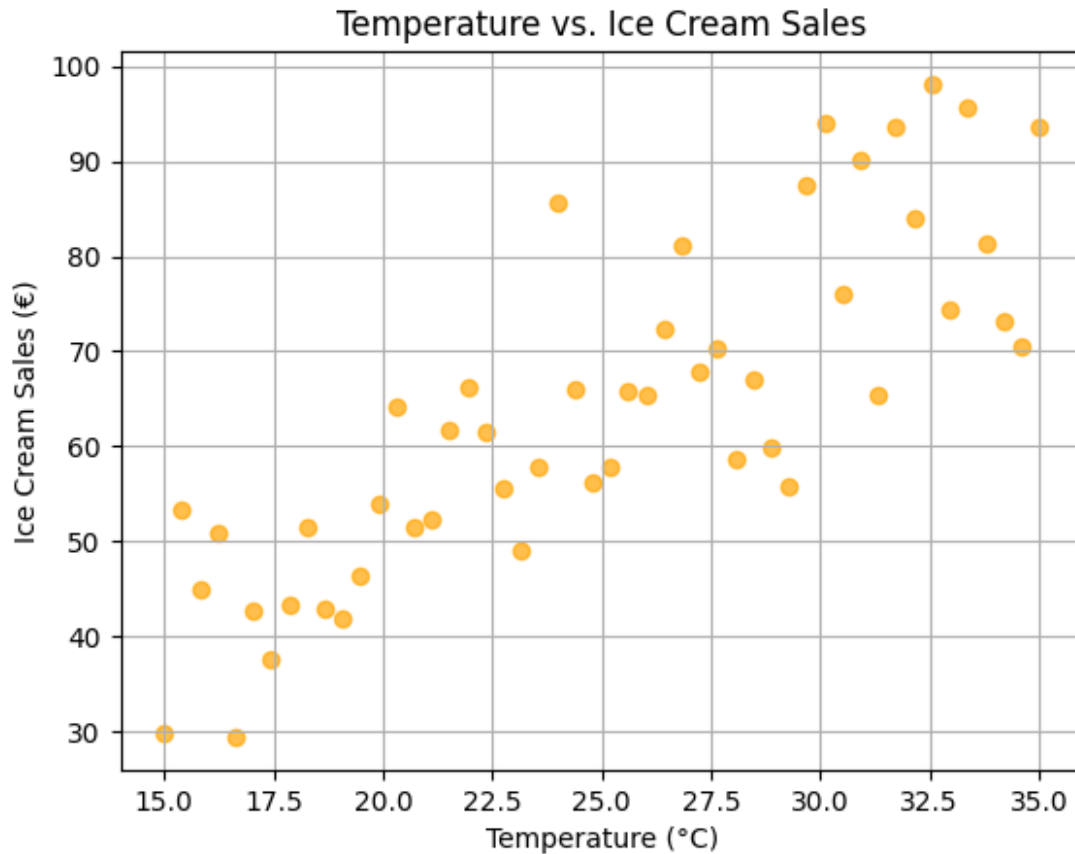
# Scatter plot
plt.scatter(height, weight, color='teal', alpha=0.7)
plt.title("Height vs. Weight")
plt.xlabel("Height (cm)")
plt.ylabel("Weight (kg)")
plt.grid(True)
plt.show()
```



□ Example 2: Temperature vs. Ice Cream Sales

```
temperature = np.linspace(15, 35, 50)
sales = 2.5 * temperature + np.random.normal(0, 10, 50)

plt.scatter(temperature, sales, color='orange', alpha=0.7)
plt.title("Temperature vs. Ice Cream Sales")
plt.xlabel("Temperature (°C)")
plt.ylabel("Ice Cream Sales (€)")
plt.grid(True)
plt.show()
```



Line graph

A line graph (also called a line chart) is a type of graph used to show how a variable changes over time or how two variables are related.

It is made up of points (data values) connected by straight lines.

Example 1: Simple Line Graph

Description: This example shows a basic line graph that visualizes a single variable changing over time — for instance, temperature over a week.

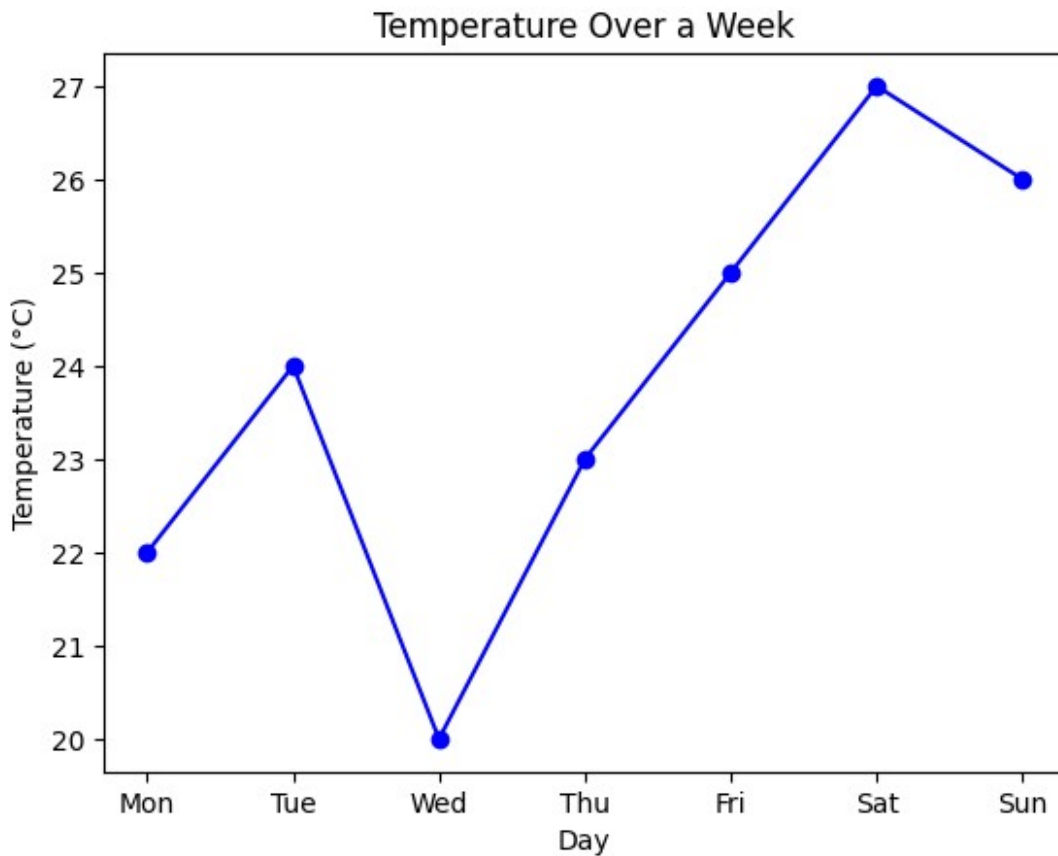
```
import matplotlib.pyplot as plt

# Data
days = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
temperature = [22, 24, 20, 23, 25, 27, 26]

# Plot
plt.plot(days, temperature, marker='o', color='b', linestyle='--')

# Labels and Title
plt.title('Temperature Over a Week')
plt.xlabel('Day')
```

```
plt.ylabel('Temperature (°C)')  
plt.show()
```



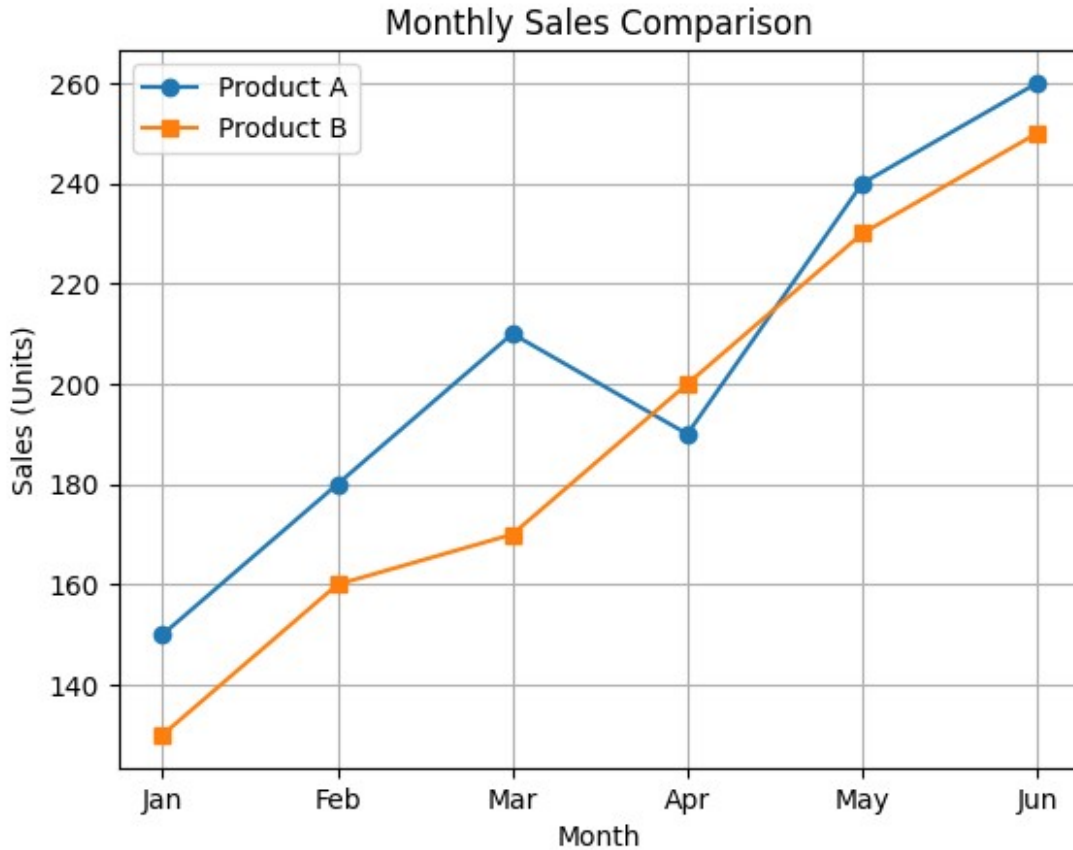
Example 2: Multiple Line Graphs

Description: This example plots two lines to compare data — for example, comparing sales for two products over months.

```
# Data  
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun']  
product_A = [150, 180, 210, 190, 240, 260]  
product_B = [130, 160, 170, 200, 230, 250]  
  
# Plot multiple lines  
plt.plot(months, product_A, marker='o', label='Product A')  
plt.plot(months, product_B, marker='s', label='Product B')  
  
# Add labels and legend  
plt.title('Monthly Sales Comparison')  
plt.xlabel('Month')
```

```
plt.ylabel('Sales (Units)')
plt.legend()
plt.grid(True)

plt.show()
```



Examples from datasets

```
df = pd.read_csv('heart.csv')
df.head()
```

slope \	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak
0 2	52	1	0	125	212	0	1	168	0	1.0
1 0	53	1	0	140	203	1	0	155	1	3.1
2 0	70	1	0	145	174	0	1	125	1	2.6
3 2	61	1	0	148	203	0	1	161	0	0.0
4 1	62	0	0	138	294	1	1	106	0	1.9

	ca	thal	target
0	2	3	0
1	0	3	0
2	0	3	0
3	1	3	0
4	3	2	0

Features of dataset

- age (#)
- sex : 1= Male, 0= Female (Binary)
- (cp)chest pain type (4 values -Ordinal):Value 1: typical angina ,Value 2: atypical angina, Value 3: non-anginal pain , Value 4: asymptomatic
- (trestbps) resting blood pressure (#)
- (chol) serum cholesterol in mg/dl (#)
- (fbs)fasting blood sugar > 120 mg/dl(Binary)(1 = true; 0 = false)
- (restecg) resting electrocardiography results(values 0,1,2)
- (thalach) maximum heart rate achieved (#)
- (exang) exercise induced angina (binary) (1 = yes; 0 = no)
- (oldpeak) = ST depression induced by exercise relative to rest (#)
- (slope) of the peak exercise ST segment (Ordinal) (Value 1: up sloping , Value 2: flat , Value 3: down sloping)
- (ca) number of major vessels (0–3, Ordinal) colored by fluoroscopy
- (thal) maximum heart rate achieved — (Ordinal): 3 = normal; 6 = fixed defect; 7 = reversible defect

df.describe()

	age	sex	cp	trestbps	chol
\					
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000
mean	54.434146	0.695610	0.942439	131.611707	246.000000
std	9.072290	0.460373	1.029641	17.516718	51.59251
min	29.000000	0.000000	0.000000	94.000000	126.000000
25%	48.000000	0.000000	0.000000	120.000000	211.000000
50%	56.000000	1.000000	1.000000	130.000000	240.000000
75%	61.000000	1.000000	2.000000	140.000000	275.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000
	fbs	restecg	thalach	exang	oldpeak

\	count	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000
mean	0.149268	0.529756	149.114146	0.336585	1.071512	
std	0.356527	0.527878	23.005724	0.472772	1.175053	
min	0.000000	0.000000	71.000000	0.000000	0.000000	
25%	0.000000	0.000000	132.000000	0.000000	0.000000	
50%	0.000000	1.000000	152.000000	0.000000	0.800000	
75%	0.000000	1.000000	166.000000	1.000000	1.800000	
max	1.000000	2.000000	202.000000	1.000000	6.200000	

	slope	ca	thal	target
count	1025.000000	1025.000000	1025.000000	1025.000000
mean	1.385366	0.754146	2.323902	0.513171
std	0.617755	1.030798	0.620660	0.500070
min	0.000000	0.000000	0.000000	0.000000
25%	1.000000	0.000000	2.000000	0.000000
50%	1.000000	0.000000	2.000000	1.000000
75%	2.000000	1.000000	3.000000	1.000000
max	2.000000	4.000000	3.000000	1.000000

Correlation heatmap

The image you provided appears to be a correlation heatmap, which visualizes the correlation coefficients between different variables in a dataset. Color Coding:

- The color gradient ranges from blue to red, indicating the strength and direction of the correlation:
 - Blue shades suggest negative correlations, meaning that as one variable increases, the other tends to decrease.
 - Red shades indicate positive correlations, meaning that as one variable increases, the other also tends to increase.
- The intensity of the color reflects the strength of the correlation: Darker shades indicate stronger correlations (close to -1 or 1), while lighter shades indicate weaker correlations (close to 0).

Correlation Coefficients:

- Each cell in the heatmap shows the correlation coefficient (ranging from -1 to 1) between the variables corresponding to that cell's row and column:
 - A value of 1 indicates a perfect positive correlation.
 - A value of -1 indicates a perfect negative correlation.

- A value of 0 indicates no correlation.

Observations:

- There is a strong negative correlation between oldpeak and thalach, suggesting that higher levels of oldpeak (depression induced by exercise) are associated with lower maximum heart rates achieved.
- The correlation between sex and chol appears to be weak.
- The variable age shows a relatively strong negative correlation with thalach, indicating that as age increases, the maximum heart rate achieved tends to decrease.

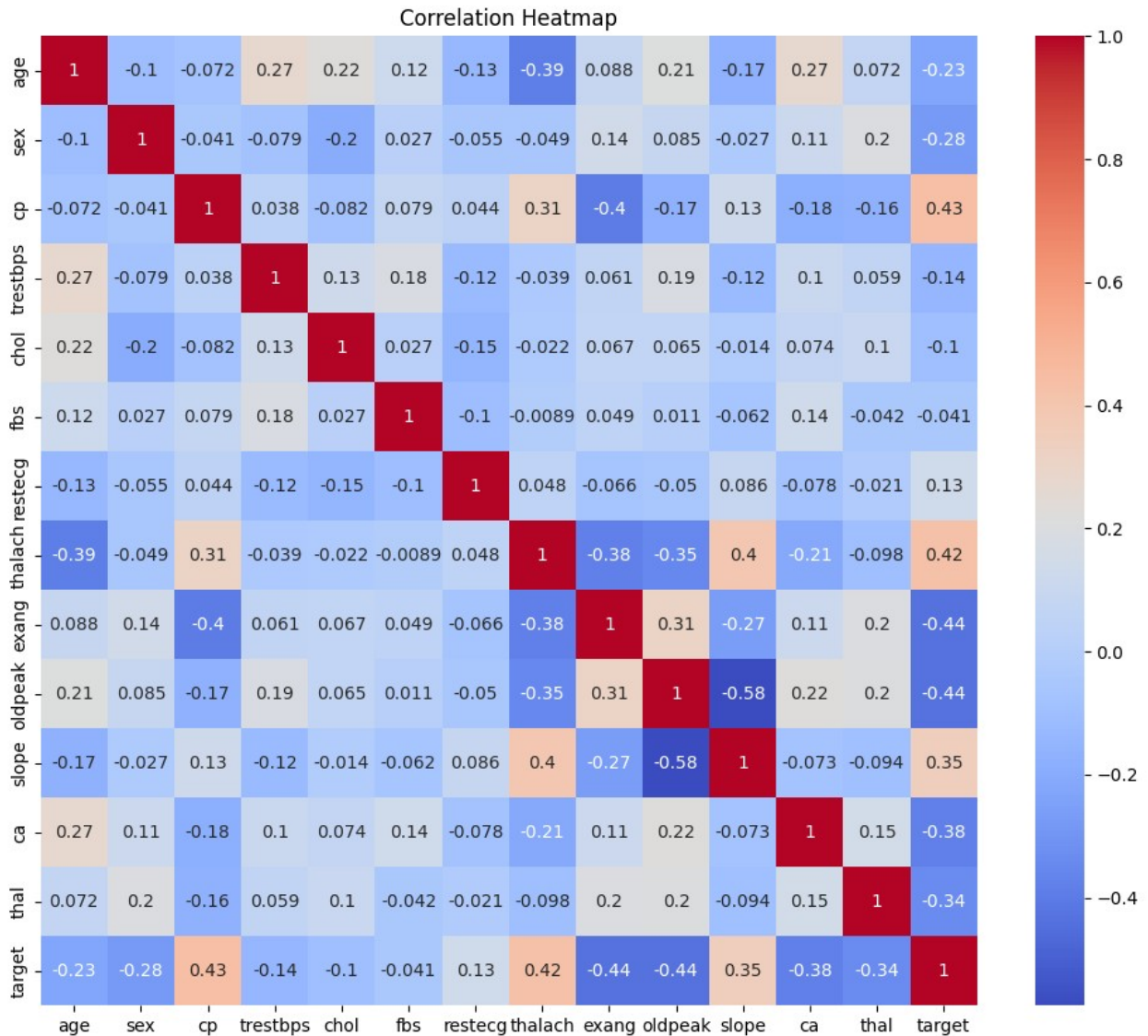
Number of males: 713

Number of females: 312

```
plt.figure(figsize=(12, 10))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

-1

1



Bar chart

```
# Check the gender column
gender_counts = df['sex'].value_counts() # 'sex' is typically coded
as 0 (female) and 1 (male)

# Display the counts
print("Number of males:", gender_counts[1]) # Assuming 1 is for male
print("Number of females:", gender_counts[0]) # Assuming 0 is for
female

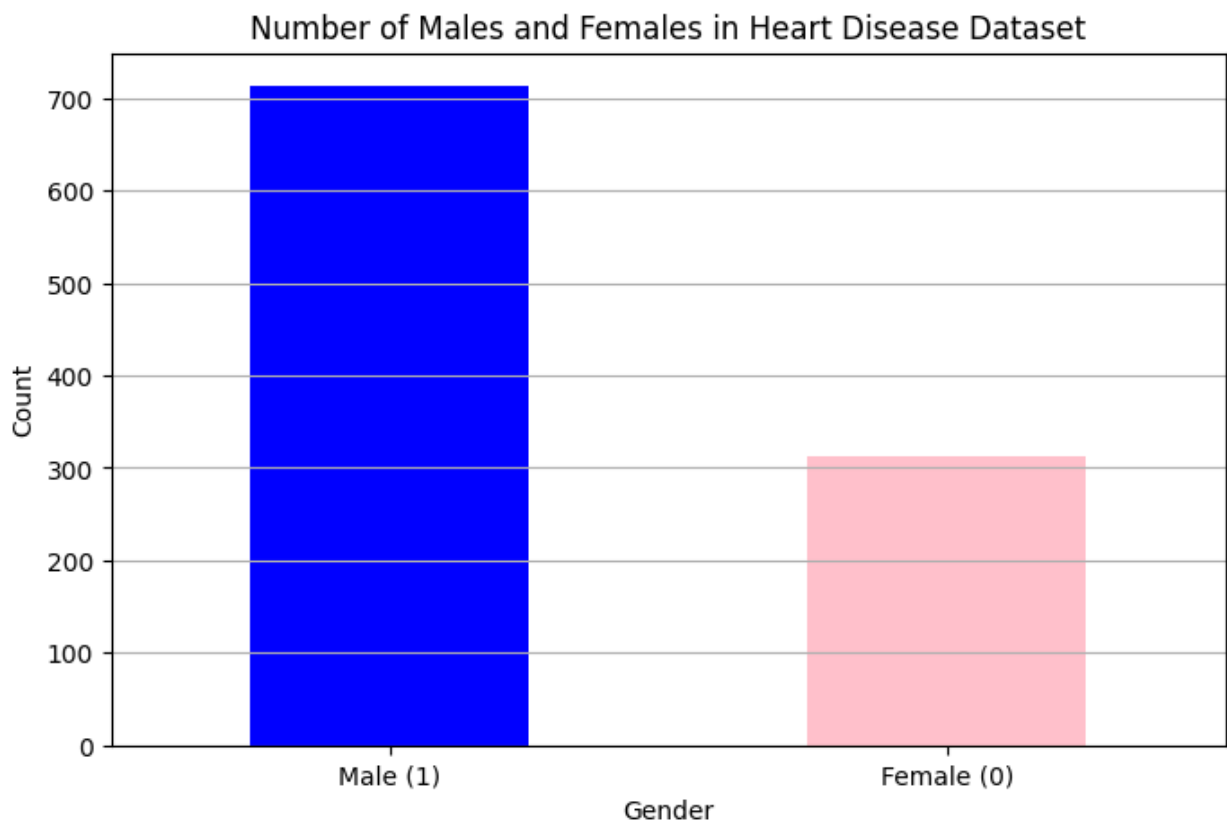
Number of males: 713
Number of females: 312

# Create a bar chart
plt.figure(figsize=(8, 5))
```

```

gender_counts.plot(kind='bar', color=['blue', 'pink'])
plt.xticks([0, 1], ['Male (1)', 'Female (0)'], rotation=0)
plt.title('Number of Males and Females in Heart Disease Dataset')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.grid(axis='y')
plt.show()

```



```

# Group by target (heart disease presence) and sex, then count
sex_class_counts = df.groupby(['target',
                                'sex']).size().unstack(fill_value=0)

```

```

# Display the result
print(sex_class_counts)

```

```

sex      0      1
target
0         86   413
1        226   300

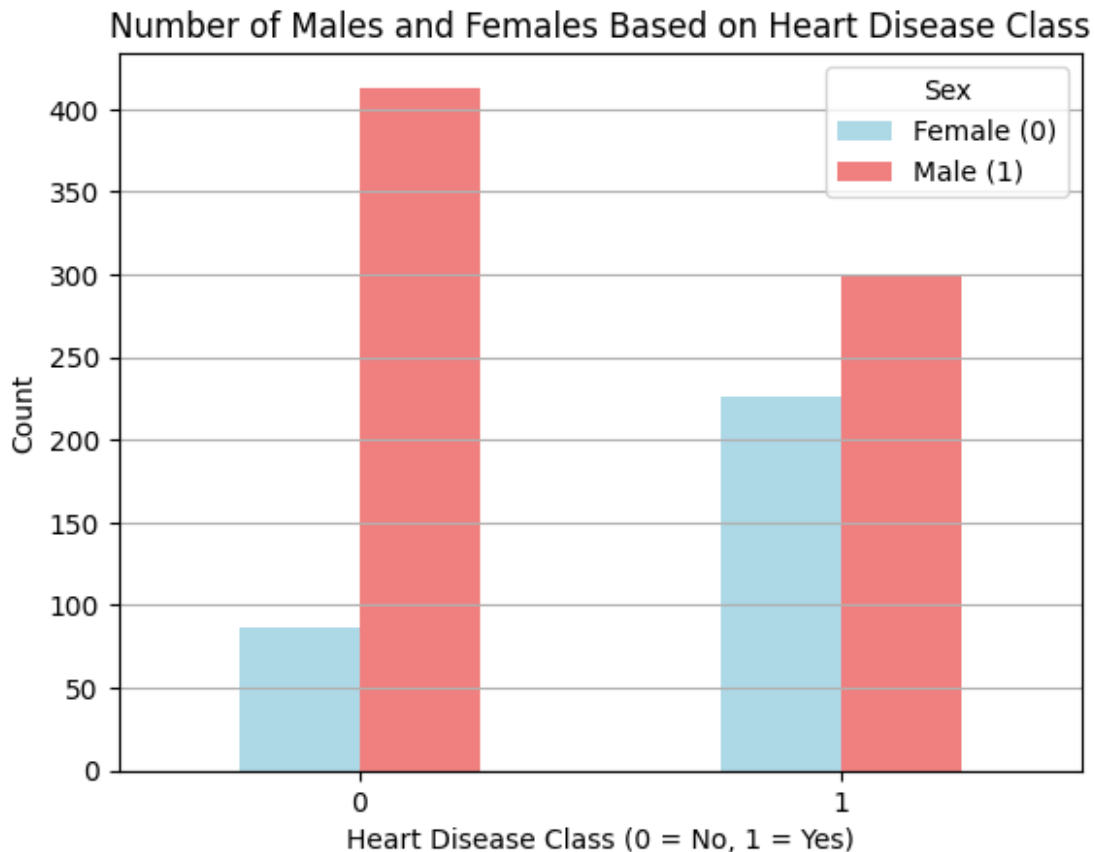
```

```

# Create a bar chart
sex_class_counts.plot(kind='bar', color=['lightblue', 'lightcoral'])
plt.title('Number of Males and Females Based on Heart Disease Class')
plt.xlabel('Heart Disease Class (0 = No, 1 = Yes)')

```

```
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.legend(title='Sex', labels=['Female (0)', 'Male (1)'])
plt.grid(axis='y')
plt.show()
```



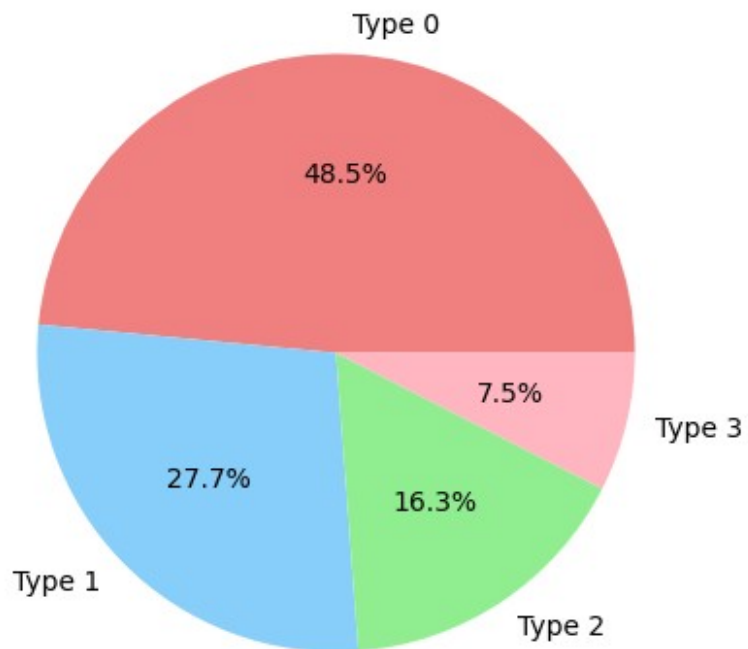
Pie chart

A pie chart is a circular statistical graphic that is divided into slices to illustrate numerical proportions. Each slice's arc length is proportional to the quantity it represents.

```
cp_counts = df['cp'].value_counts()

# Create a pie chart
plt.figure(figsize=(8, 5))
plt.pie(cp_counts, labels=['Type 0', 'Type 1', 'Type 2', 'Type 3'],
        autopct='%1.1f%%', colors=['lightcoral', 'lightskyblue', 'lightgreen',
        'lightpink'])
plt.title('Distribution of Chest Pain Types (cp)')
plt.show()
```

Distribution of Chest Pain Types (cp)

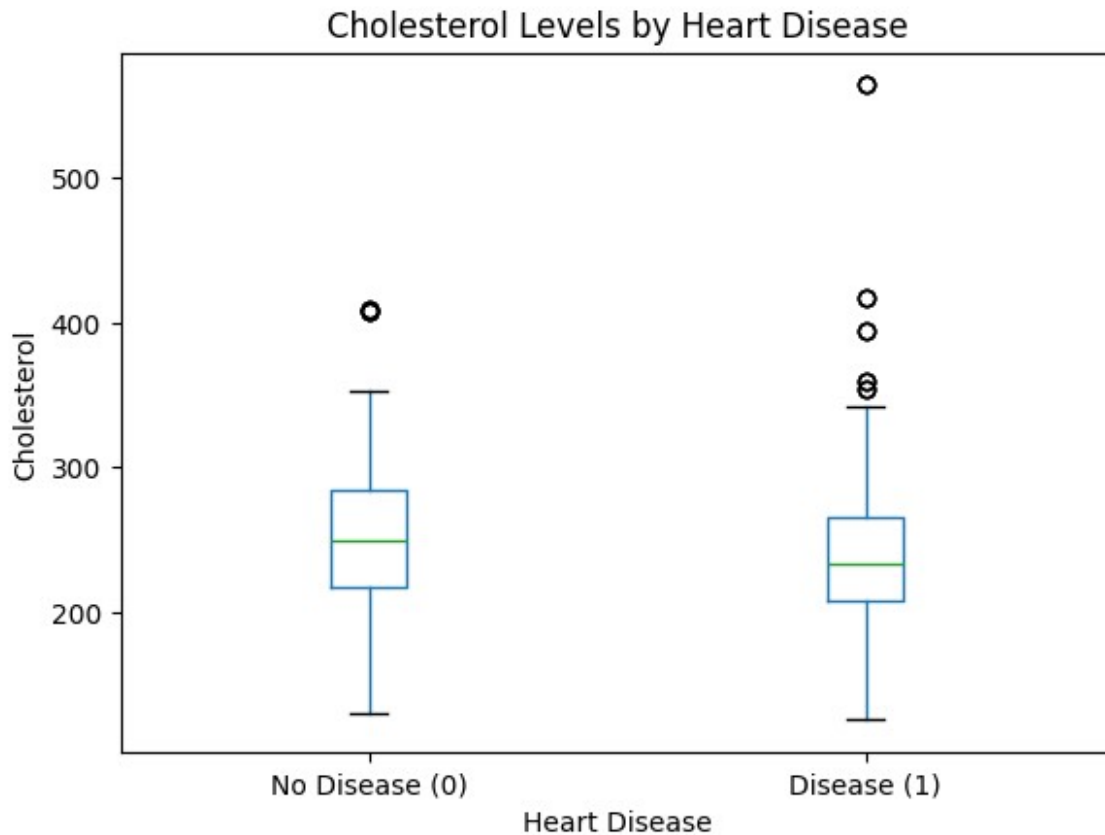


Box Plot

Cholesterol levels are often associated with heart disease risk. We can visualize the distribution of cholesterol levels among patients with and without heart disease using a box plot.

```
# Create the box plot
plt.figure(figsize=(8, 5))
df.boxplot(column='chol', by='target', grid=False)
plt.title('Cholesterol Levels by Heart Disease')
plt.suptitle('')
plt.xlabel('Heart Disease')
plt.ylabel('Cholesterol')
plt.xticks([1, 2], ['No Disease (0)', 'Disease (1)'])
plt.show()
```

<Figure size 800x500 with 0 Axes>



Line graph

A line graph (or line chart) is a type of chart used to visualize data points that are connected by straight lines.

Line graph presents the average of cholesterol levels by Age. The x-axis represents age (starting from around 30 years and increasing up to about 75), while the y-axis represents the average cholesterol level, with values ranging from 200 to 300.

```
age_grouped = df.groupby('age')['chol'].mean().reset_index()

# Create a line chart
plt.figure(figsize=(10, 6))
plt.plot(age_grouped['age'], age_grouped['chol'], marker='o',
         linestyle='-', color='blue')
plt.title('Average Cholesterol Levels by Age')
plt.xlabel('Age')
plt.ylabel('Average Cholesterol Level')
plt.grid()
plt.show()
```

