Matplotlib

Matplotlib is a **Python library** used for creating static and interactive visualizations.

- `pyplot` – A module in Matplotlib that provides a MATLAB-like interface for simple plotting.

🎯 Features:

1. **Data Visualization:**
   - Helps in visualizing large datasets, making it easier to identify trends, patterns, and correlations.
   - Converts raw data into intuitive and informative graphs.

2. **Customization:**
   - Highly customizable with options to modify colors, labels, legends, and styles to create professional-looking plots.

3. **Supports Multiple Plot Types:**
   - Line plots, bar charts, histograms, scatter plots, pie charts, box plots, and more.

4. **Integration with Other Libraries:**
   - Works seamlessly with libraries like **NumPy**, **Pandas**, and **Seaborn** for data manipulation and visualization.

Syntax to import the library

```
import matplotlib.pyplot as plt
```

Matplotlib Plotting Functions: Definitions and Syntax

plot()

Used to create line plots by plotting `x` and `y` coordinates.

Syntax:

```
plt.plot(x, y, linestyle, marker, color, label)
```

Parameters:

- x and y – Data points for the X and Y axes.
- linestyle – Type of line (e.g., '-', '--', ':').
- marker – Style of markers (e.g., 'o', '^', '*').
- color – Line color (e.g., 'red', 'blue').
- label – Name of the line (used with legend()).

Matplotlib Linestyle, Marker, and Color Reference Table

1. Linestyle Options

Symbol	Description
'-'	Solid line
'--'	Dashed line
'-.'	Dash-dot line
:	Dotted line
'' or ' '	No line (invisible)

3. Color Options

Symbol	Color Name	Hex Code
b	Blue	#0000FF
g	Green	#008000
r	Red	#FF0000
c	Cyan	#00FFFF
m	Magenta	#FF00FF
y	Yellow	#FFFF00
k	Black	#000000
w	White	#FFFFFF

2. Marker Options

Symbol	Description
.	Point marker
,	Pixel marker
o	Circle marker
v	Triangle down marker
^	Triangle up marker
<	Triangle left marker
>	Triangle right marker
1	Tri-down marker
2	Tri-up marker
3	Tri-left marker
4	Tri-right marker
s	Square marker
p	Pentagon marker
*	Star marker
h	Hexagon 1 marker
H	Hexagon 2 marker
+	Plus marker
x	X marker
D	Diamond marker
d	Thin diamond marker
|	Vertical line marker
_	Horizontal line marker

figure()

Creates a new figure or activates an existing one to plot multiple plots.

Syntax:

```
plt.figure(figsize=(width, height))
```

Parameters:

`figsize` – Tuple defining width and height in inches.

title()

Adds a title to the plot.

Syntax:

```
plt.title('Your Title', fontsize=value, color='color')
```

xlabel()

Adds a label to the X-axis.

Syntax:

```
plt.xlabel('X-axis Label', fontsize=value, color='color')
```

ylabel()

Adds a label to the Y-axis.

Syntax:

```
plt.ylabel('Y-axis Label', fontsize=value, color='color')
```

xticks()

Sets the locations and labels of the X-axis ticks.

Syntax:

```
plt.xticks(tick_values, tick_labels, rotation=value)
```

yticks()

Sets the locations and labels of the Y-axis ticks.

Syntax:

```
plt.yticks(tick_values, tick_labels, rotation=value)
```

legend()

Adds a legend to the plot to show labels of plotted data.

Syntax:

```
plt.legend(loc='position')
```

grid()

Adds a grid to the plot for better readability.

Syntax:

```
plt.grid(True, linestyle='--', color='gray')
```

show()

Displays the plot.

Syntax:

```
plt.show()
```

subplot()

Creates multiple plots in a grid layout within a single figure.

Syntax:

```
plt.subplot(nrows, ncols, index)
```

subplots()

Creates multiple subplots and returns a figure and axes object.

Syntax:

```
fig, ax = plt.subplots(nrows, ncols, figsize=(width, height))
```

suptitle()

Adds a title for the entire figure when using subplots.

Syntax:

```
plt.suptitle('Main Title', fontsize=value, color='color')
```

set_xlabel()

Sets the label for the X-axis on a specific subplot.

Syntax:

```
ax.set_xlabel('X-axis Label')
```

set_ylabel()

Sets the label for the Y-axis on a specific subplot.

Syntax:

```
ax.set_ylabel('Y-axis Label')
```

tight_layout()

automatically adjusts the spacing between subplots to prevent overlapping and make the layout look better.

Syntax:

```
plt.tight_layout(rect)
```

Matplotlib Examples

1. Example Using subplot()

```
# Importing required libraries
import matplotlib.pyplot as plt
import numpy as np

# Data for plotting
x = np.linspace(0, 10, 50)
y1 = np.sin(x)
y2 = np.cos(x)

# ---- First Plot: Sine Wave ----
plt.subplot(2, 1, 1)
plt.plot(x, y1, linestyle='-', marker='o', color='b', label='Sine Wave')
plt.title('Sine Wave')
plt.xlabel('X-axis (Time)')
plt.ylabel('Y-axis (Amplitude)')
plt.xticks(np.arange(0, 11, 2))
plt.yticks(np.arange(-1, 1.5, 0.5))
plt.legend(loc='upper right')
plt.grid(True)

# ---- Second Plot: Cosine Wave ----
plt.subplot(2, 1, 2)
plt.plot(x, y2, linestyle='--', marker='s', color='r', label='Cosine Wave')
plt.title('Cosine Wave')
plt.xlabel('X-axis (Time)')
plt.ylabel('Y-axis (Amplitude)')
plt.xticks(np.arange(0, 11, 2))
plt.yticks(np.arange(-1, 1.5, 0.5))
plt.legend(loc='lower right')
plt.grid(True)

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

2. Example Using subplots()

```
# Importing required libraries
import matplotlib.pyplot as plt
import numpy as np

# Data for plotting
x = np.linspace(0, 10, 50)
y1 = np.sin(x)
y2 = np.cos(x)

# Create figure and axes
fig, ax = plt.subplots(2, 1, figsize=(8, 6))

# ---- First Plot: Sine Wave ----
ax[0].plot(x, y1, linestyle='-', marker='o', color='b', label='Sine Wave')
ax[0].set_title('Sine Wave')
ax[0].set_xlabel('X-axis (Time)')
ax[0].set_ylabel('Y-axis (Amplitude)')
ax[0].set_xticks(np.arange(0, 11, 2))
ax[0].set_yticks(np.arange(-1, 1.5, 0.5))
ax[0].legend(loc='upper right')
ax[0].grid(True)

# ---- Second Plot: Cosine Wave ----
ax[1].plot(x, y2, linestyle='--', marker='s', color='r', label='Cosine
Wave')
ax[1].set_title('Cosine Wave')
ax[1].set_xlabel('X-axis (Time)')
ax[1].set_ylabel('Y-axis (Amplitude)')
ax[1].set_xticks(np.arange(0, 11, 2))
ax[1].set_yticks(np.arange(-1, 1.5, 0.5))
ax[1].legend(loc='lower right')
ax[1].grid(True)

plt.suptitle('Sine and Cosine Wave Plots', fontsize=14, color='green')
plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show()
```

Seaborn

Seaborn is a Python data visualization library built on top of **Matplotlib**

Seaborn works well with **Pandas DataFrames** and provides built-in functions to handle datasets, making data visualization simpler and more intuitive

Syntax:

```
pip install seaborn

import seaborn as sns
```

Why use plots/graphs?

- Makes data easier to understand.
- Helps identify trends, patterns, and outliers.
- Simplifies complex data by presenting it visually.
- Supports better decision-making with clear insights.

In data visualization, plots are often classified into **Univariate**, **Bivariate**, and **Multivariate** based on the number of variables they represent.

1) **Univariate:** Focuses on one variable .

- Count Plot
- KDE Plot
- Histplot
- Pie Chart

2) **Bivariate:** Focuses on the relationship between two variables .

- Scatter Plot
- Line Plot
- Bar Plot
- Box Plot
- Lmplot (Linear Model Plot)
- Jointplot

3) **Multivariate:** Focuses on the relationship between three or more variables .

- Pair Plot

- Heatmap
- FacetGrid

1) Count Plot:

- **What:** Displays the count of each category in a column.
- **When:** Use it to see how often each category appears.
- **Why:** Quickly understand the distribution of categorical data.

Syntax:
```
sns.countplot(data=df  , x='column_name' , hue)
```

2) KDE Plot (Kernel Density Estimate):

- **What:** Shows the smooth distribution of a numeric variable.
- **When:** Use it to understand the shape of the data (e.g., peaks, spread, skewness).
- **Why:** Helps visualize the probability density of the data.

Syntax:
```
sns.kdeplot(x='column_name',data=df ,hue,fill=True)
```

3) Histplot (Histogram Plot):

- **What:** Shows the distribution of numeric data by dividing it into bins.
- **When:** Use it to see the frequency of values in intervals.
- **Why:** Helps understand data spread and identify patterns.

Syntax:
```
sns.histplot(x='column_name',data=df,kde=True,hue)
```

4) Pie Chart:

- **What:** Shows the proportion of each category as slices of a circle.
- **When:** Use it to compare parts of a whole.
- **Why:** Helps visualize percentage or ratio distribution.

Syntax:
```
plt.pie(values, labels=categories,
           Autopct , explode )
```

5) Scatter Plot:

- **What:** Shows the relationship between two numeric variables using points.
- **When:** Use it to identify patterns, trends, or correlations.
- **Why:** Helps visualize how one variable affects another.

Syntax:
```
sns.scatterplot(x='column1', y='column2', data=df,
                            Hue , style , size)
```

6) Line Plot:

- **What:** Shows the relationship between two numeric variables using a continuous line.
- **When:** Use it to track changes over time or a sequence.
- **Why:** Helps identify trends or patterns.

Syntax:
```
sns.lineplot(x='column1', y='column2', data=df)
```

7) Bar Plot:

- **What:** Shows the relationship between a categorical variable and a numeric variable using bars.
- **When:** Use it to compare values across different categories.
- **Why:** Helps visualize differences between groups.

Syntax:
```
sns.barplot(x='category_column', y='value_column',
                  data=df , estimator='mean', hue)
```

```
estimator can be mean , median , size(count) , max , min etc..
```

8) Box Plot:

- **What:** Displays the distribution of numeric data and identifies outliers.
- **When:** Use it to understand data spread, median, and detect outliers.
- **Why:** Helps visualize data distribution and variability.

Syntax:
```
sns.boxplot(x='category_column', y='value_column',
                  data=df , hue)
```

9) Lmplot (Linear Model Plot):

- **What:** Shows the relationship between two numeric variables with a linear regression line.
- **When:** Use it to visualize data points and fit a regression line.
- **Why:** Helps understand trends and relationships.

Syntax:
```
sns.lmplot(x='column1', y='column2', data=df)
```

10) Jointplot:

- **What:** Combines scatter plot and distribution plots (kde / histplot) to show the relationship between two numeric variables.
- **When:** Use it to analyze correlation along with individual distributions.
- **Why:** Provides a detailed view of bivariate relationships and marginal distributions.

Syntax: `sns.jointplot(x='column1', y='column2', data=df)`

11) Heatmap:

- **What:** Displays data in a matrix form using color to represent values.
- **When:** Use it to show correlations, patterns, or intensity in data.
- **Why:** Helps quickly spot trends and relationships.

Syntax: `sns.heatmap(dataframe_corr, annot=True, cmap)`

12) Pairplot:

- **What:** Plots pairwise relationships between numeric variables in a dataset.
- **When:** Use it to explore relationships between multiple variables at once.
- **Why:** Helps identify patterns and correlations across several features.

Syntax: `sns.pairplot(df)`

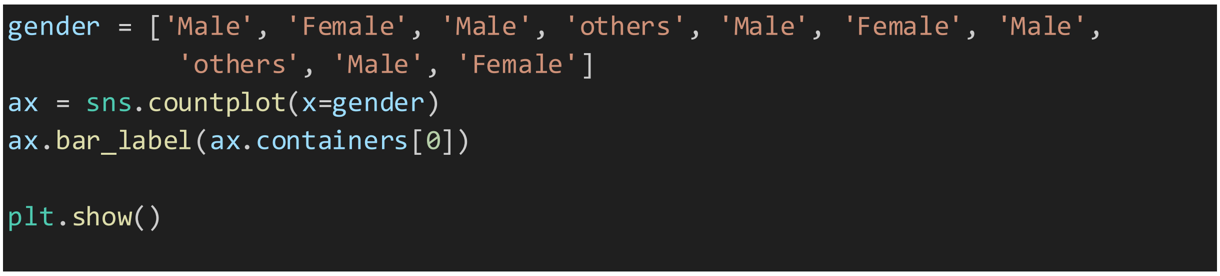
13) FacetGrid:

- **What:** Creates a grid of plots based on categorical variables.
- **When:** Use it to visualize data across multiple subsets.
- **Why:** Helps compare distributions or relationships across categories.

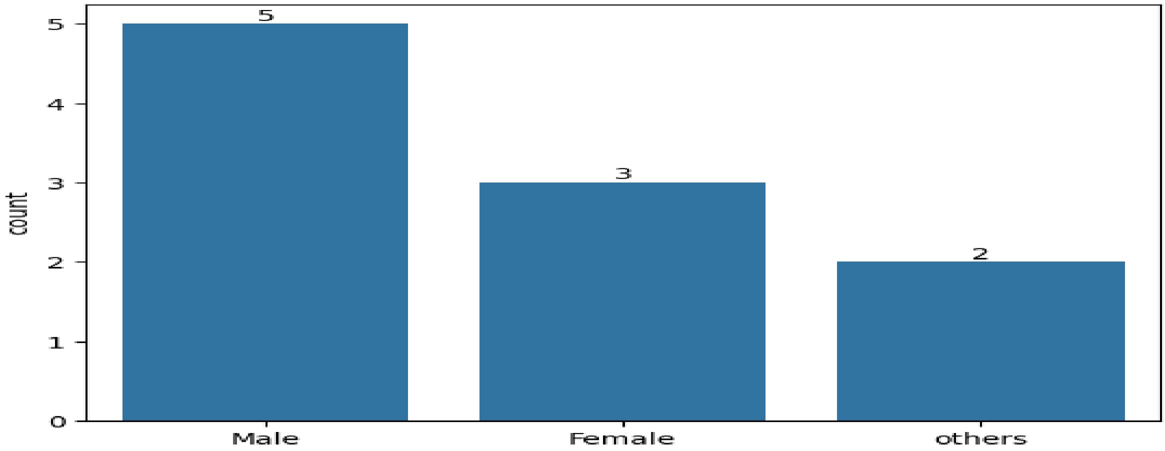
Syntax:

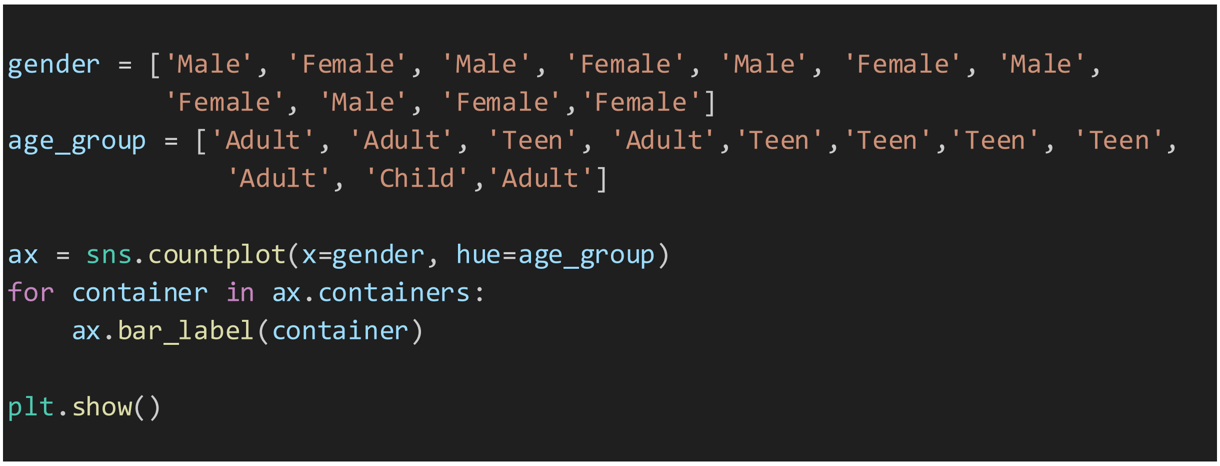
```
g = sns.FacetGrid(df, col='category_column')

g.map(sns.scatterplot, 'column1', 'column2')
```

countplot

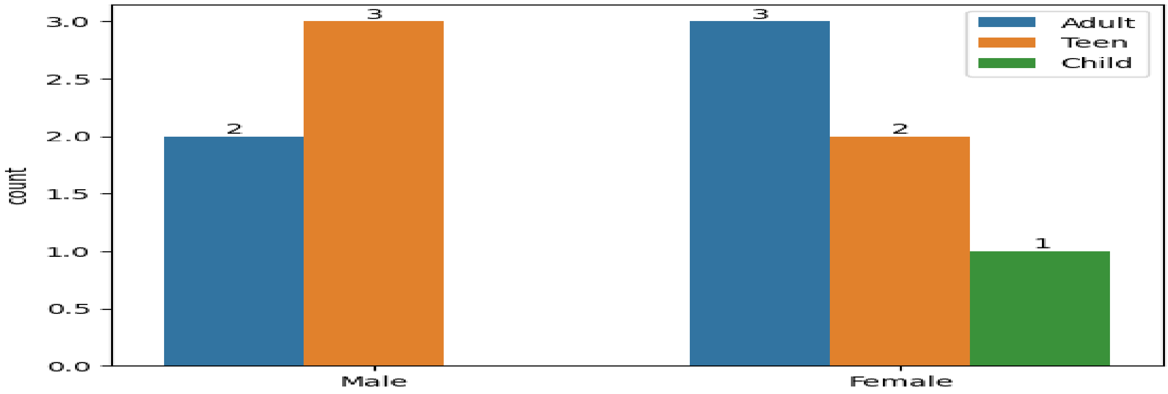


```
gender = ['Male', 'Female', 'Male', 'others', 'Male', 'Female', 'Male',
           'others', 'Male', 'Female']
ax = sns.countplot(x=gender)
ax.bar_label(ax.containers[0])

plt.show()
```

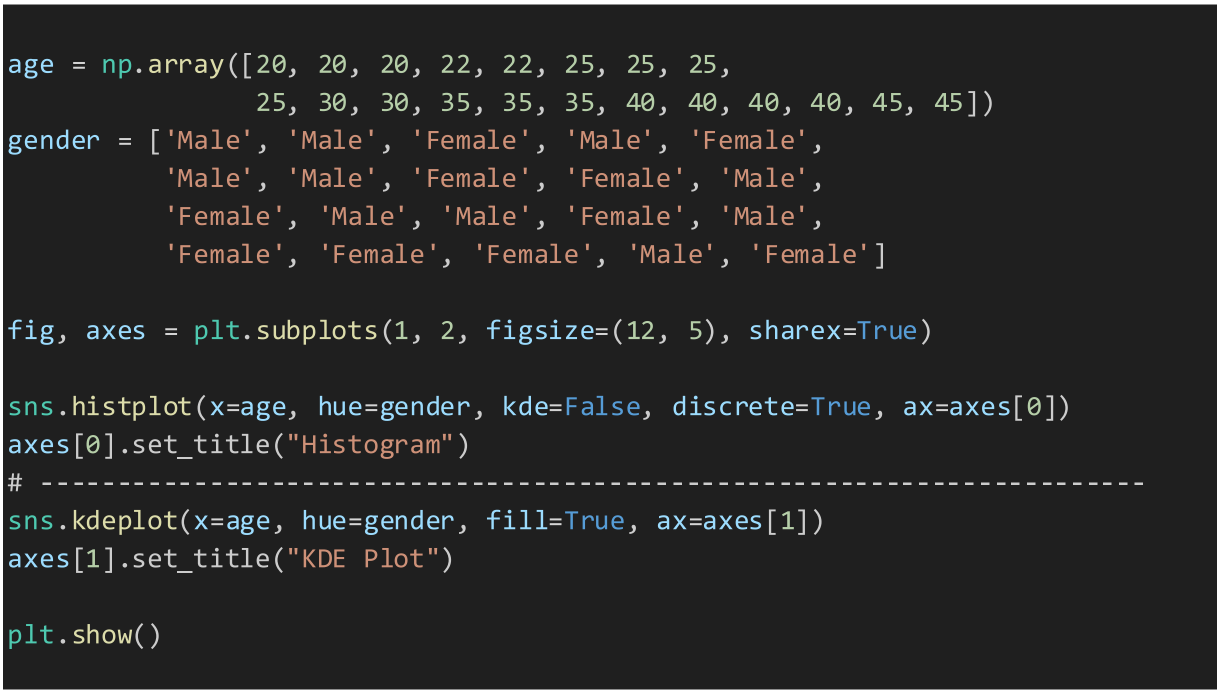




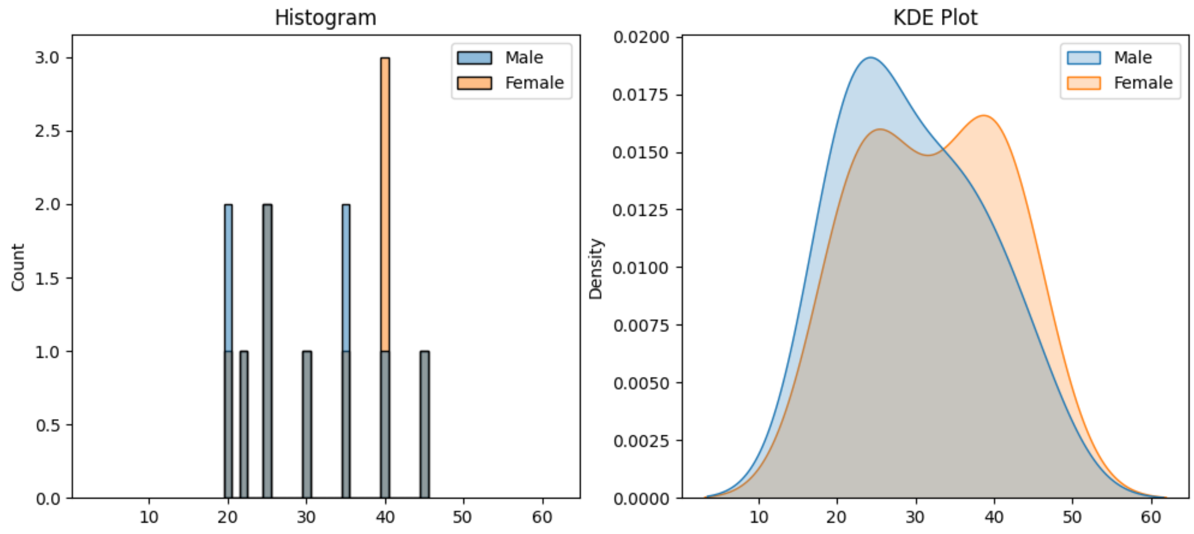
```
gender = ['Male', 'Female', 'Male', 'Female', 'Male', 'Female', 'Male',
          'Female', 'Male', 'Female','Female']
age_group = ['Adult', 'Adult', 'Teen', 'Adult','Teen','Teen','Teen', 'Teen',
              'Adult', 'Child','Adult']

ax = sns.countplot(x=gender, hue=age_group)
for container in ax.containers:
    ax.bar_label(container)

plt.show()
```



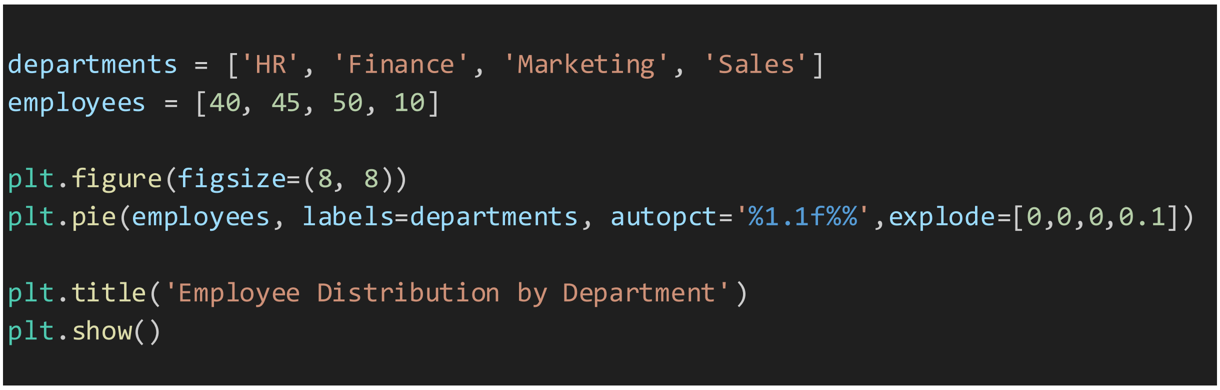
kdeplot and histplot



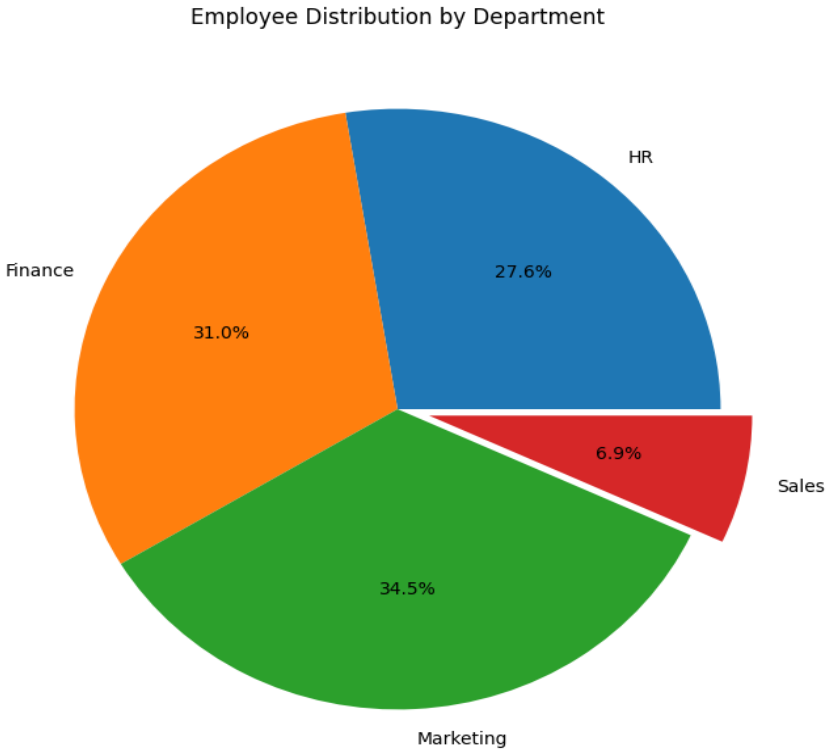
```
age = np.array([20, 20, 20, 22, 22, 25, 25, 25,
                25, 30, 30, 35, 35, 35, 40, 40, 40, 40, 45, 45])
gender = ['Male', 'Male', 'Female', 'Male', 'Female',
          'Male', 'Male', 'Female', 'Female', 'Male',
          'Female', 'Male', 'Male', 'Female', 'Male',
          'Female', 'Female', 'Female', 'Male', 'Female']

fig, axes = plt.subplots(1, 2, figsize=(12, 5), sharex=True)

sns.histplot(x=age, hue=gender, kde=False, discrete=True, ax=axes[0])
axes[0].set_title("Histogram")
# -----------------------------------------------------------------------
sns.kdeplot(x=age, hue=gender, fill=True, ax=axes[1])
axes[1].set_title("KDE Plot")

plt.show()
```



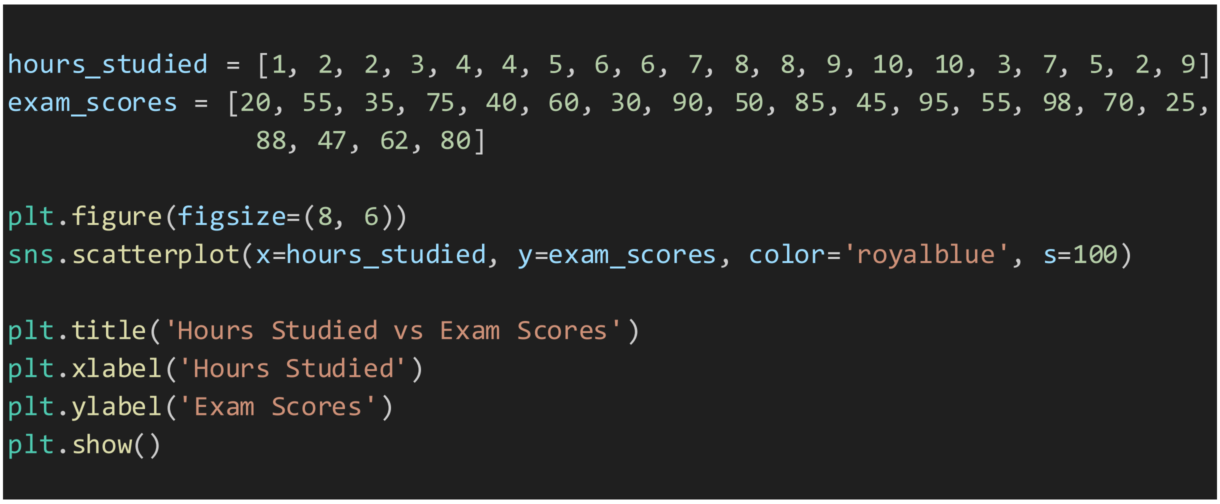
pie chart



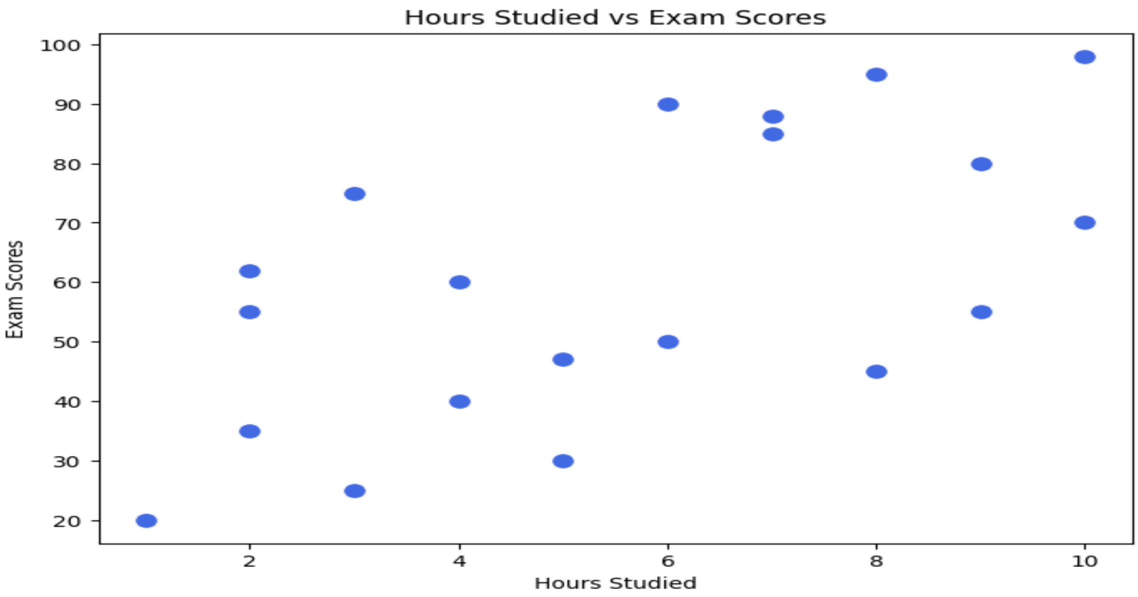
```
departments = ['HR', 'Finance', 'Marketing', 'Sales']
employees = [40, 45, 50, 10]

plt.figure(figsize=(8, 8))
plt.pie(employees, labels=departments, autopct='%1.1f%%',explode=[0,0,0,0.1])

plt.title('Employee Distribution by Department')
plt.show()
```

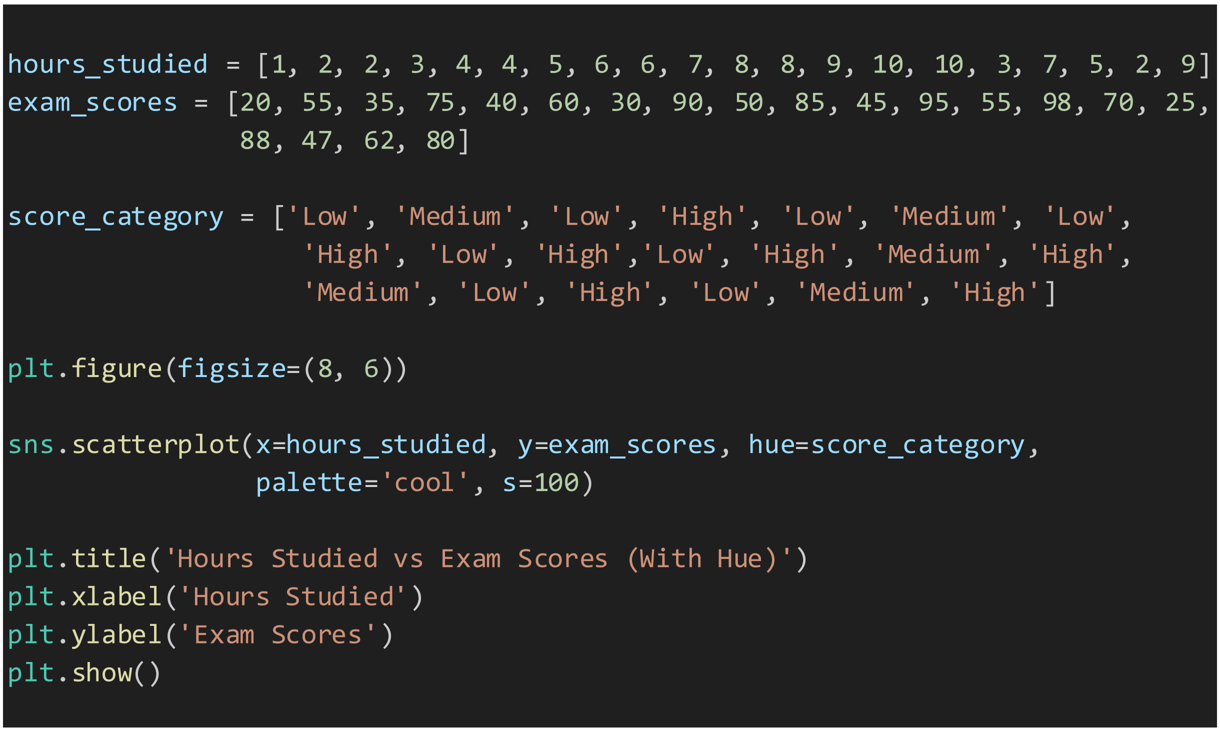


Scatterplot

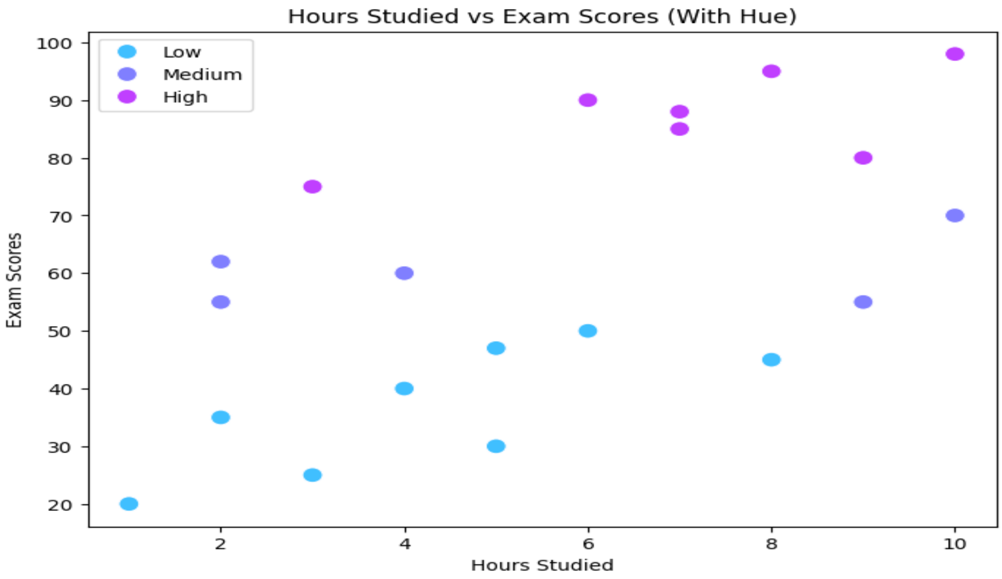


```
hours_studied = [1, 2, 2, 3, 4, 4, 5, 6, 6, 7, 8, 8, 9, 10, 10, 3, 7, 5, 2, 9]
exam_scores = [20, 55, 35, 75, 40, 60, 30, 90, 50, 85, 45, 95, 55, 98, 70, 25,
                88, 47, 62, 80]

plt.figure(figsize=(8, 6))
sns.scatterplot(x=hours_studied, y=exam_scores, color='royalblue', s=100)

plt.title('Hours Studied vs Exam Scores')
plt.xlabel('Hours Studied')
plt.ylabel('Exam Scores')
plt.show()
```

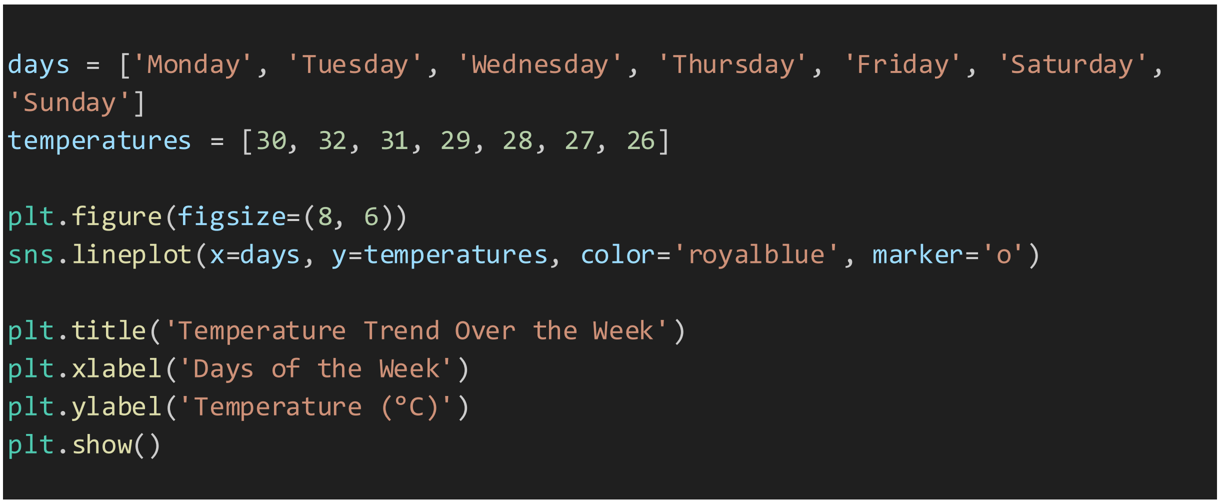




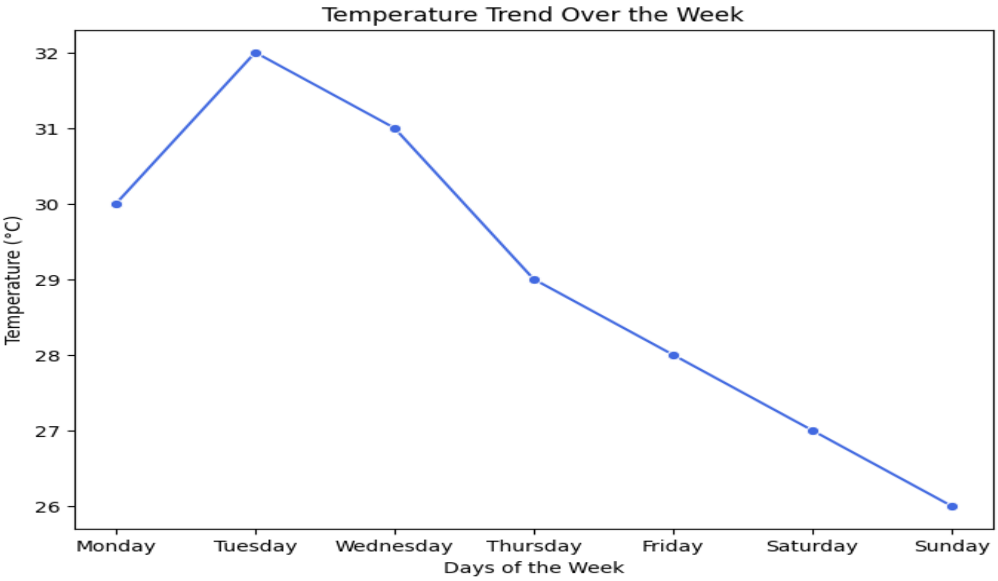
```
hours_studied = [1, 2, 2, 3, 4, 4, 5, 6, 6, 7, 8, 8, 9, 10, 10, 3, 7, 5, 2, 9]
exam_scores = [20, 55, 35, 75, 40, 60, 30, 90, 50, 85, 45, 95, 55, 98, 70, 25,
               88, 47, 62, 80]

score_category = ['Low', 'Medium', 'Low', 'High', 'Low', 'Medium', 'Low',
                  'High', 'Low', 'High','Low', 'High', 'Medium', 'High',
                  'Medium', 'Low', 'High', 'Low', 'Medium', 'High']

plt.figure(figsize=(8, 6))

sns.scatterplot(x=hours_studied, y=exam_scores, hue=score_category,
                palette='cool', s=100)

plt.title('Hours Studied vs Exam Scores (With Hue)')
plt.xlabel('Hours Studied')
plt.ylabel('Exam Scores')
plt.show()
```



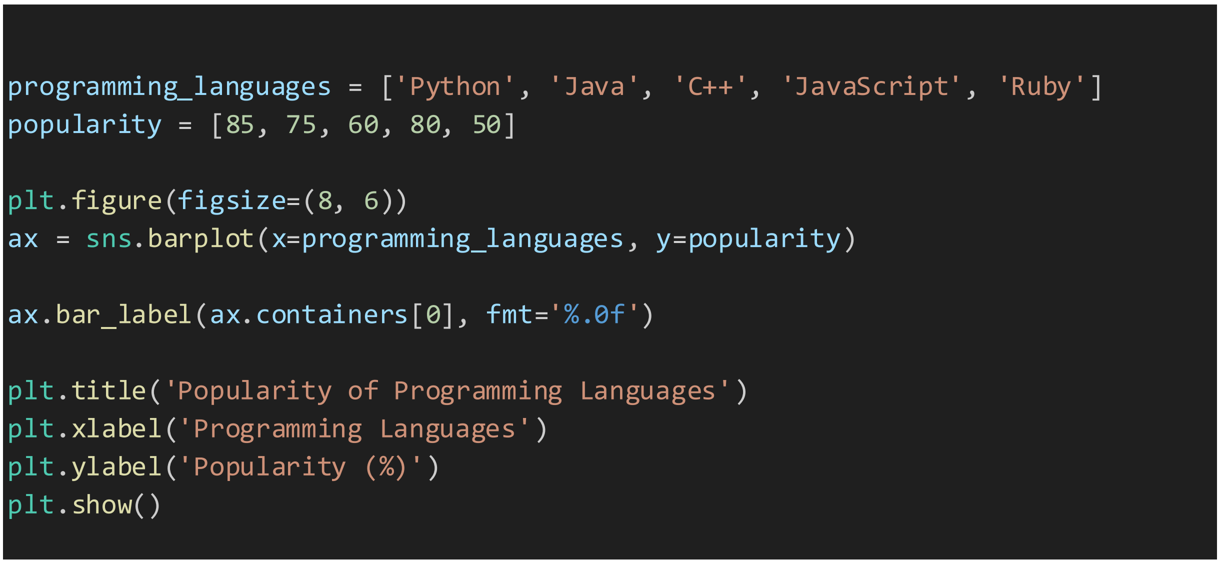
lineplot



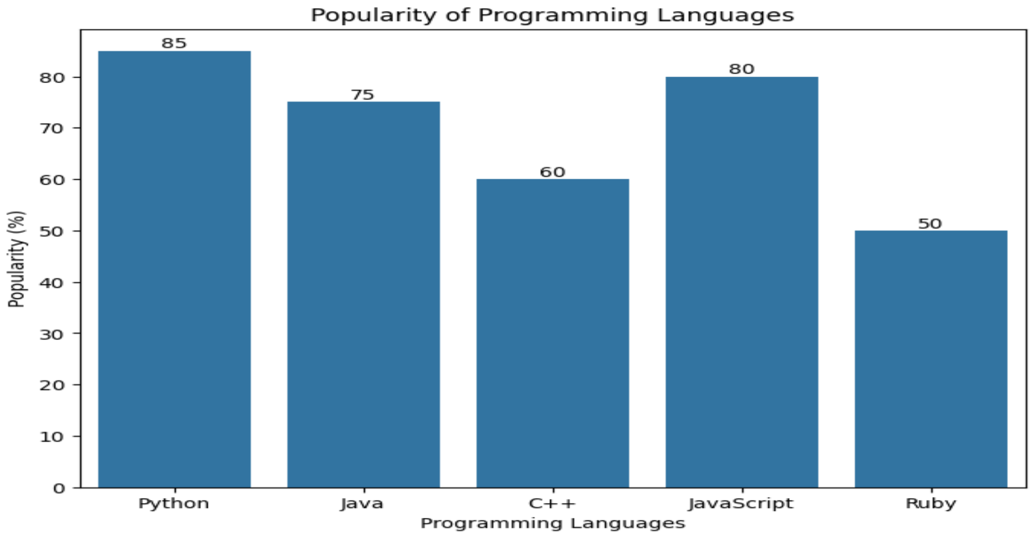
```
days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday',
'Sunday']
temperatures = [30, 32, 31, 29, 28, 27, 26]

plt.figure(figsize=(8, 6))
sns.lineplot(x=days, y=temperatures, color='royalblue', marker='o')

plt.title('Temperature Trend Over the Week')
plt.xlabel('Days of the Week')
plt.ylabel('Temperature (°C)')
plt.show()
```



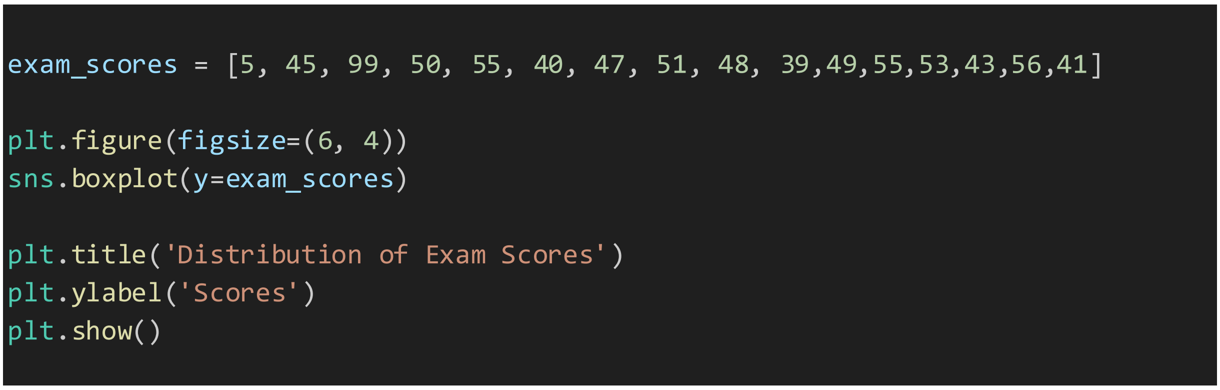
barplot



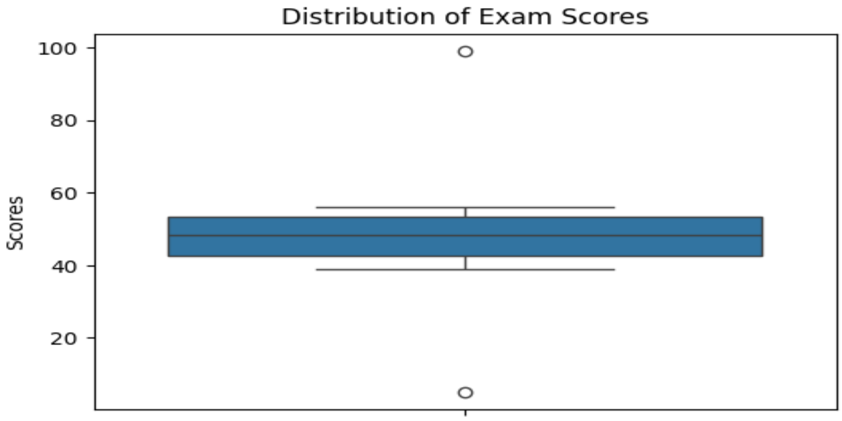
```
programming_languages = ['Python', 'Java', 'C++', 'JavaScript', 'Ruby']
popularity = [85, 75, 60, 80, 50]

plt.figure(figsize=(8, 6))
ax = sns.barplot(x=programming_languages, y=popularity)

ax.bar_label(ax.containers[0], fmt='%.0f')

plt.title('Popularity of Programming Languages')
plt.xlabel('Programming Languages')
plt.ylabel('Popularity (%)')
plt.show()
```

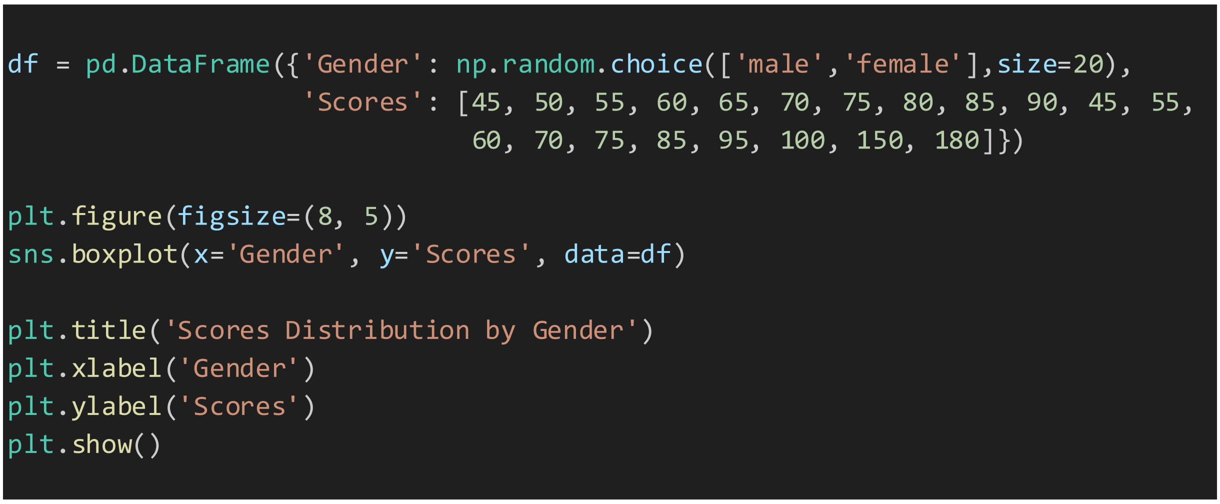


boxplot

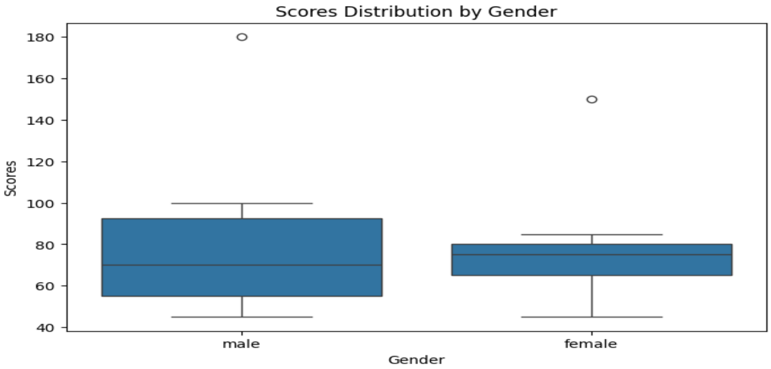


```
exam_scores = [5, 45, 99, 50, 55, 40, 47, 51, 48, 39,49,55,53,43,56,41]

plt.figure(figsize=(6, 4))
sns.boxplot(y=exam_scores)

plt.title('Distribution of Exam Scores')
plt.ylabel('Scores')
plt.show()
```



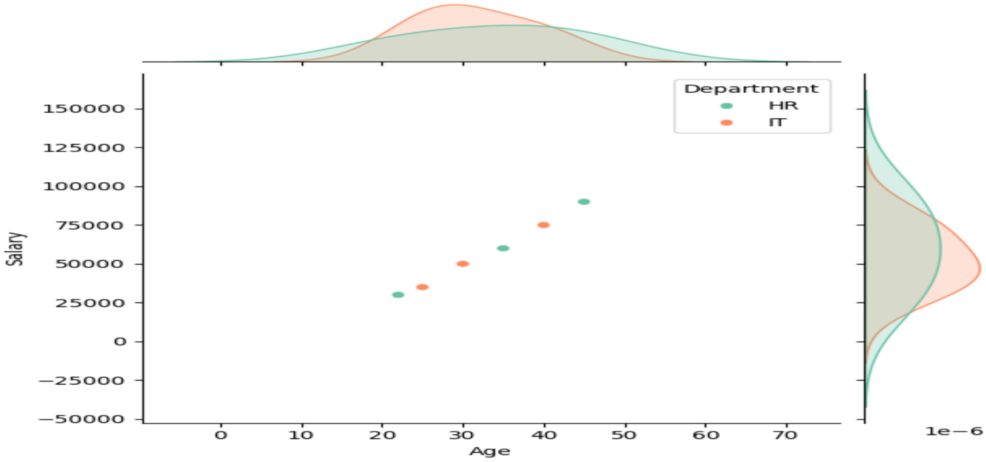


```
df = pd.DataFrame({'Gender': np.random.choice(['male','female'],size=20),
                   'Scores': [45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 45, 55,
                              60, 70, 75, 85, 95, 100, 150, 180]})

plt.figure(figsize=(8, 5))
sns.boxplot(x='Gender', y='Scores', data=df)

plt.title('Scores Distribution by Gender')
plt.xlabel('Gender')
plt.ylabel('Scores')
plt.show()
```



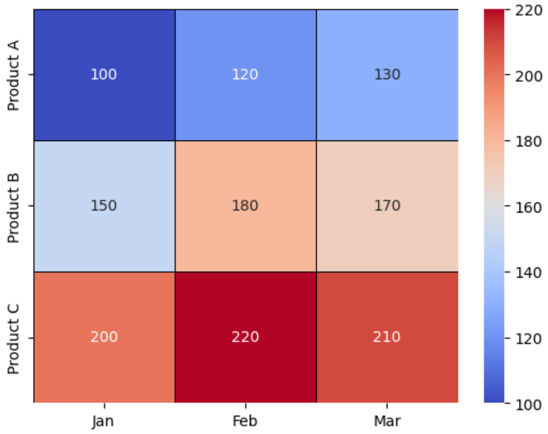
jointplot

```
df = pd.DataFrame({'Age': [22, 25, 30, 35, 40, 45],
                   'Salary':  [30000, 35000, 50000, 60000, 75000, 90000] ,
                   'Department':['HR', 'IT', 'IT', 'HR', 'IT', 'HR']})

sns.jointplot(data=df, x='Age', y='Salary', hue='Department', palette='Set2')

plt.show()
```



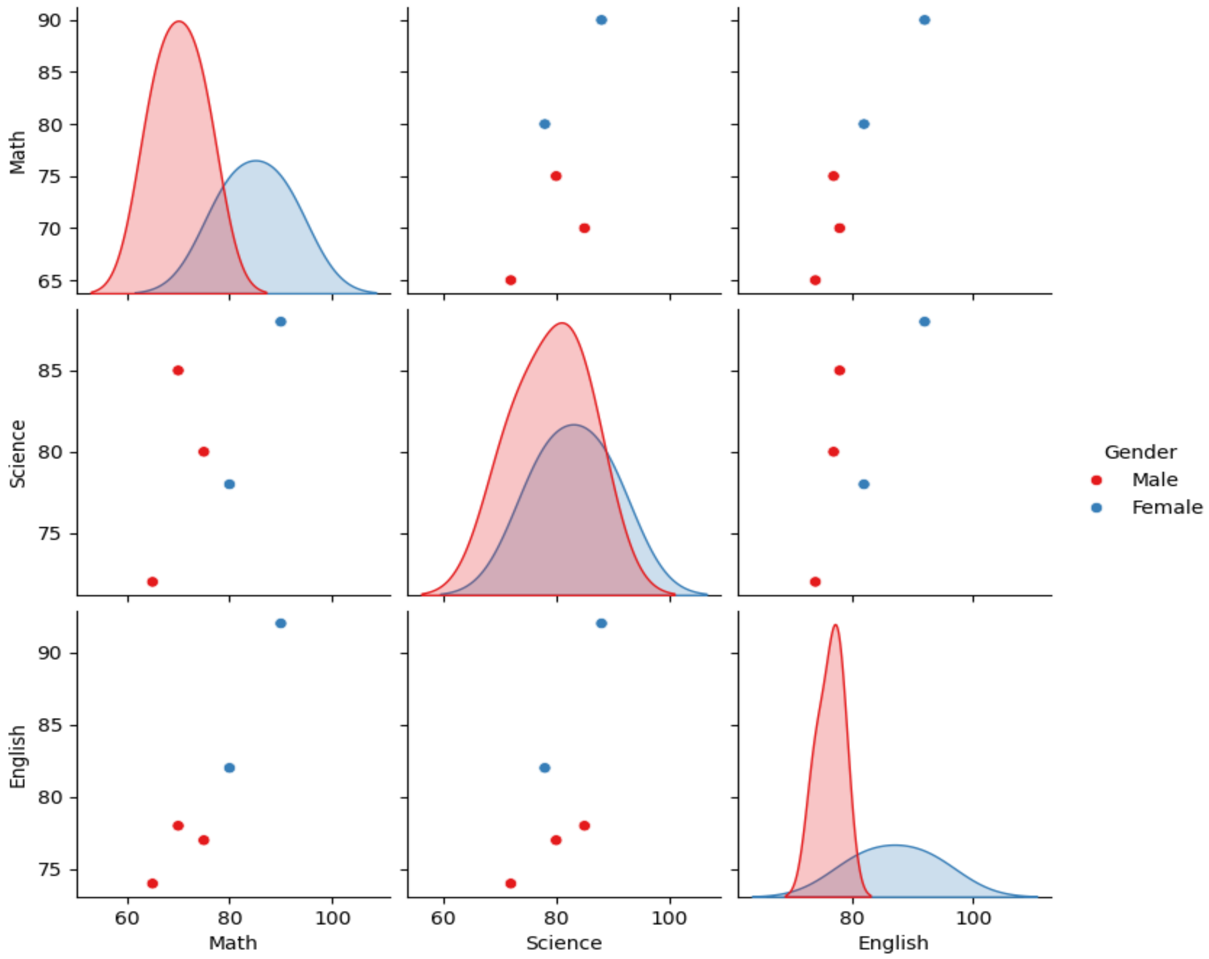
heatmap

```
df = pd.DataFrame({  'Jan': [100, 150, 200],
                     'Feb': [120, 180, 220],
                      'Mar': [130, 170, 210] },
                  index=['Product A', 'Product B', 'Product C'])

sns.heatmap(df, annot=True, cmap='coolwarm', fmt='d', linewidths=0.5,
linecolor='black')
plt.show()
```

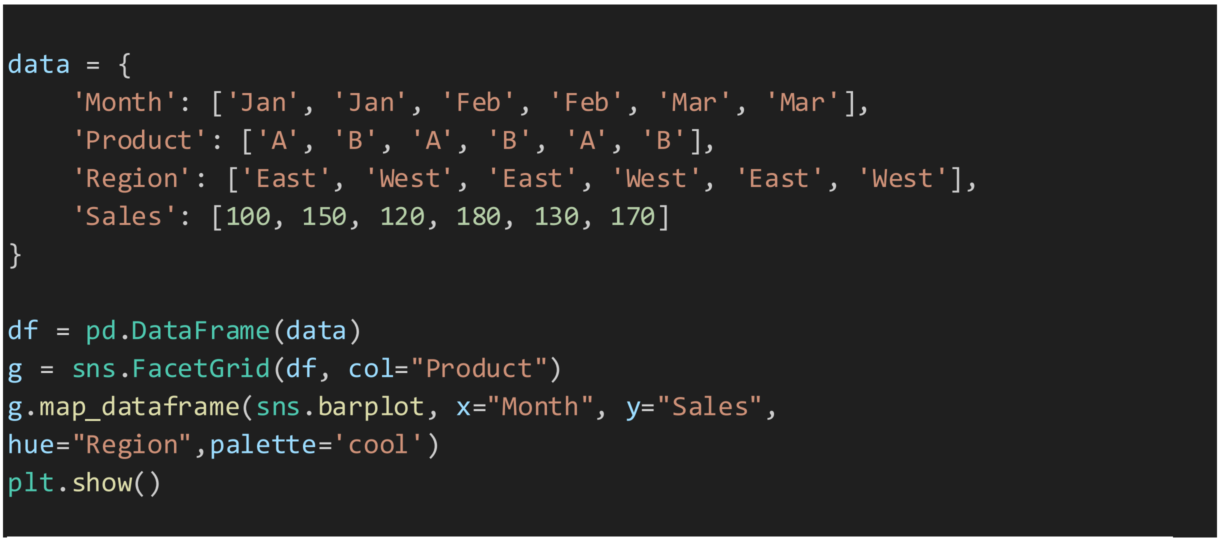


pairplot

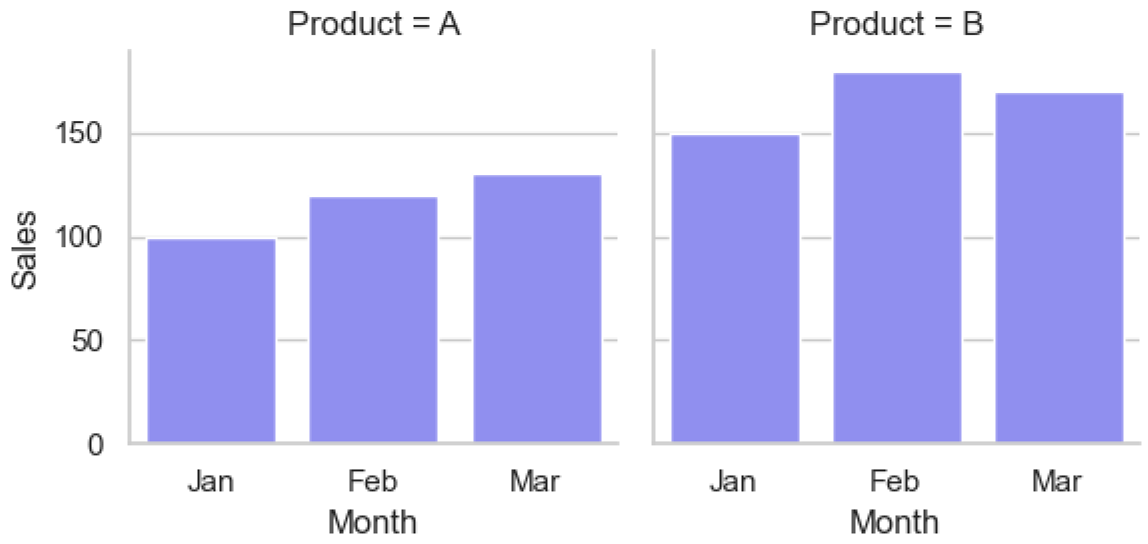
```
data = {
    'Math': [70, 80, 65, 90, 75],
    'Science': [85, 78, 72, 88, 80],
    'English': [78, 82, 74, 92, 77],
    'Gender': ['Male', 'Female', 'Male', 'Female', 'Male']
}
df = pd.DataFrame(data)
sns.pairplot(df, hue='Gender', palette='Set1')
plt.show()
```



FacetGrid



```
data = {
    'Month': ['Jan', 'Jan', 'Feb', 'Feb', 'Mar', 'Mar'],
    'Product': ['A', 'B', 'A', 'B', 'A', 'B'],
    'Region': ['East', 'West', 'East', 'West', 'East', 'West'],
    'Sales': [100, 150, 120, 180, 130, 170]
}

df = pd.DataFrame(data)
g = sns.FacetGrid(df, col="Product")
g.map_dataframe(sns.barplot, x="Month", y="Sales",
hue="Region",palette='cool')
plt.show()
```


Product = A
Product = B
Sales
Jan
Feb
Mar
Month
Jan
Feb
Mar
Month