VISUALIZATION

In Python, data analysis and visualization are commonly performed using the Pandas and Matplotlib libraries. Pandas is used for data manipulation and analysis, while Matplotlib is mainly used for data visualization. These libraries often work together to provide comprehensive tools for data handling and graphical representation.

PANDAS

Pandas, introduced in 2008 by Wes McKinney, is an open-source Python library built on top of NumPy. It provides data structures and functions for efficiently handling structured data, making it a powerful tool for data analysis.

Key Concepts in Pandas

1. **Series**: A one-dimensional labeled array capable of holding any data type.
2. **DataFrame**: A two-dimensional labeled data structure with columns of potentially different types.
3. **Index**: Labels used to identify rows in Series or DataFrame.

Basic Functions

- `pd.read_csv()`: Load data from CSV file.
- `DataFrame.head()`: Display first 5 rows of DataFrame.
- `DataFrame.info()`: Summary of DataFrame.
- `DataFrame.describe()`: Statistical description of data.
- `DataFrame.groupby()`: Grouping data for aggregation.
- `DataFrame.merge()`: Combining two DataFrames.

Example Code

```
# Import pandas

import pandas as pd


# ---- Creating a DataFrame ----

data = {

  'Name': ['Alice', 'Bob', 'Charlie', 'David'],
```

```
    'Age': [25, 30, 35, 40],
    'City': ['New York', 'Los Angeles', 'Chicago', 'Houston'],
    'Salary': [50000, 60000, 70000, 80000]
}

df = pd.DataFrame(data)

# Display DataFrame
print("DataFrame:\n", df, "\n")

# ---- Viewing Data ----
print("First 2 rows:\n", df.head(2), "\n")
print("Last 2 rows:\n", df.tail(2), "\n")
print("Column Names:\n", df.columns, "\n")

# ---- Selecting Data ----
print("Selecting 'Name' column:\n", df['Name'], "\n")
print("Selecting multiple columns:\n", df[['Name', 'City']], "\n")

# ---- Filtering Rows ----
print("Employees older than 30:\n", df[df['Age'] > 30], "\n")

# ---- Adding a New Column ----
df['Bonus'] = df['Salary'] * 0.10
print("DataFrame with Bonus:\n", df, "\n")

# ---- Sorting ----
sorted_df = df.sort_values(by='Salary', ascending=False)
```

```
print("Sorted by Salary (descending):\n", sorted_df, "\n")

# ---- Grouping ----
avg_salary = df.groupby('City')['Salary'].mean()
print("Average Salary by City:\n", avg_salary, "\n")

# ---- Descriptive Statistics ----
print("Statistics:\n", df.describe(), "\n")
```

OUTPUT

DataFrame:

	Name	Age	City	Salary
0	Alice	25	New York	50000
1	Bob	30	Los Angeles	60000
2	Charlie	35	Chicago	70000
3	David	40	Houston	80000

First 2 rows:

	Name	Age	City	Salary
0	Alice	25	New York	50000
1	Bob	30	Los Angeles	60000

Last 2 rows:

	Name	Age	City	Salary
2	Charlie	35	Chicago	70000
3	David	40	Houston	80000

Column Names:

Index(['Name', 'Age', 'City', 'Salary'], dtype='object')

Selecting 'Name' column:

0 Alice

1 Bob

2 Charlie

3 David

Name: Name, dtype: object

Selecting multiple columns:

	Name	City
0	Alice	New York
1	Bob	Los Angeles
2	Charlie	Chicago
3	David	Houston

Employees older than 30:

	Name	Age	City	Salary
2	Charlie	35	Chicago	70000
3	David	40	Houston	80000

DataFrame with Bonus:

	Name	Age	City	Salary	Bonus
0	Alice	25	New York	50000	5000.0
1	Bob	30	Los Angeles	60000	6000.0
2	Charlie	35	Chicago	70000	7000.0
3	David	40	Houston	80000	8000.0

Sorted by Salary (descending):

	Name	Age	City	Salary	Bonus
3	David	40	Houston	80000	8000.0
2	Charlie	35	Chicago	70000	7000.0
1	Bob	30	Los Angeles	60000	6000.0
0	Alice	25	New York	50000	5000.0

Average Salary by City:

City	
Chicago	70000.0
Houston	80000.0
Los Angeles	60000.0
New York	50000.0

Name: Salary, dtype: float64

Statistics:

	Age	Salary	Bonus
count	4.000000	4.000000	4.000000
mean	32.500000	65000.000000	6500.000000
std	6.454972	12909.944487	1290.994449
min	25.000000	50000.000000	5000.000000
25%	28.750000	57500.000000	5750.000000
50%	32.500000	65000.000000	6500.000000
75%	36.250000	72500.000000	7250.000000
max	40.000000	80000.000000	8000.000000

Advantages of Pandas

- Easy handling of missing data.
- Powerful data grouping and aggregation.
- Seamless integration with other libraries.
- High performance for large datasets.

MATPLOTLIB

Matplotlib, introduced in 2002 by John Hunter. It is for plotting 2D plots which is built on numpy. A plot of matplotlib contains:

1 Figure

2 Axes

3 Axis

4 Artists

Figure

Figure is the container in which plots are present. It can have a single plot or multiple plots.

Axes

Axes are the plots, and they are artist attached to the figure. A plot can have 2 or 3 axes. set_xlabel(), set_ylabel() are the functions used to set the labels of x and y respectively.

Axis

It is responsible for generating ticks or the limits on the axes.

Artists

The whatever content visible in a figure are the artists.

PYPLOT

Pyplot is a sub library of matplotlib where all the utilities lie under. It has different types of plots including bar graphs, scatter plots, pie charts, histograms, area charts.

We import pyplot from matplotlib as following:

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

We also import numpy as a support for the matplotlib :

```
import numpy as np
```

Some of the plots in matplotlib:

LINE CHART

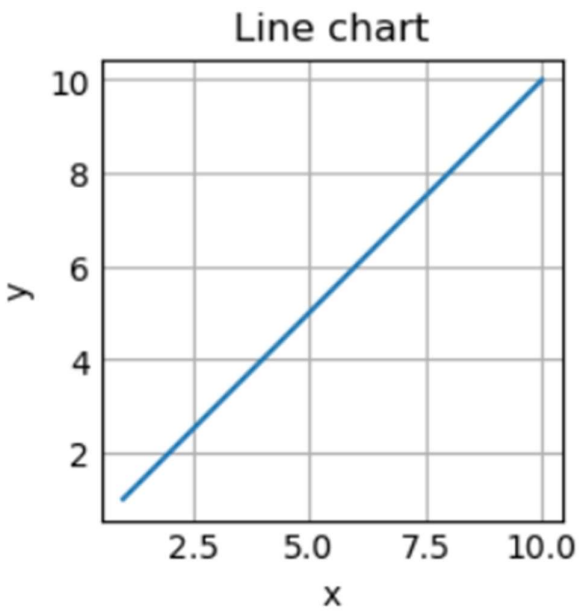
Line plots are the basic charts where dot consecutive points are connected by a continuous line.

The default plot() is used for line charts.

Code:

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3
4  x=np.array([1,10])
5  y=np.array([1,10])
6
7  plt.plot(x,y)
8  plt.title("Line chart")
9  plt.xlabel("x")
10 plt.ylabel("y")
11 plt.show()
12
```

Output:



Description:

np.array() will generate an array in the specified range.

plot() for the plotting the line chart.

title() for defining the title, ylabel() and xlabel () for labelling y and x axis respectively.

show() displays the graph.

BAR GRAPH

Rectangular bars are used in bar chart as the height represent the frequency of a particular element. bar() is used for plotting bar graphs, it has parameters like x, y, width, color.

bar(x, y, color, width)

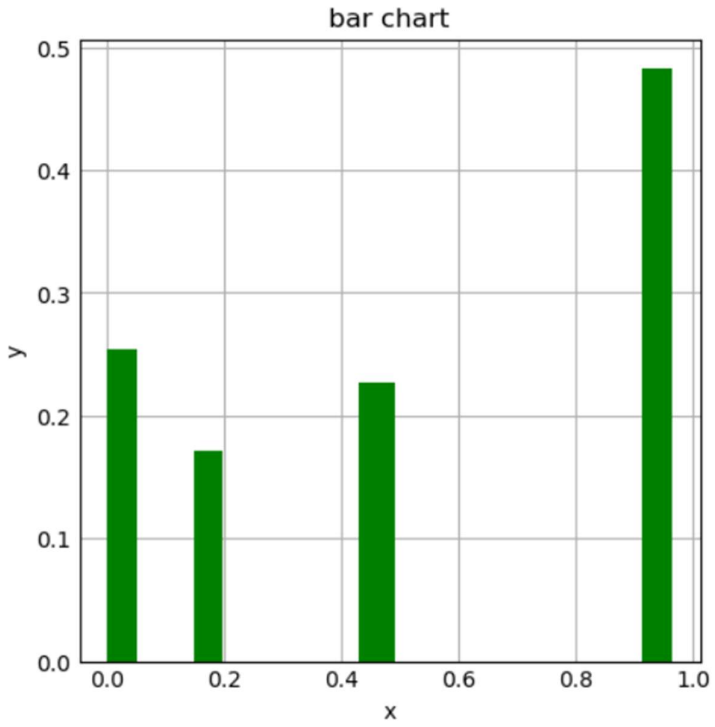
Code:

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

x=np.random.rand(5)
y=np.random.rand(5)
print(x)

fig = plt.figure(figsize = (4, 4))
plt.bar(x,y, width=0.05, color="green")
plt.title("bar chart")
plt.xlabel("x")
plt.ylabel("y")
plt.show()
```

Output:



Description:

random.rand(5) generates a random array.

Here bar() is used along with the parameters, width which denoted the width of the rectangular bars and color for the bars.

HISTOGRAM

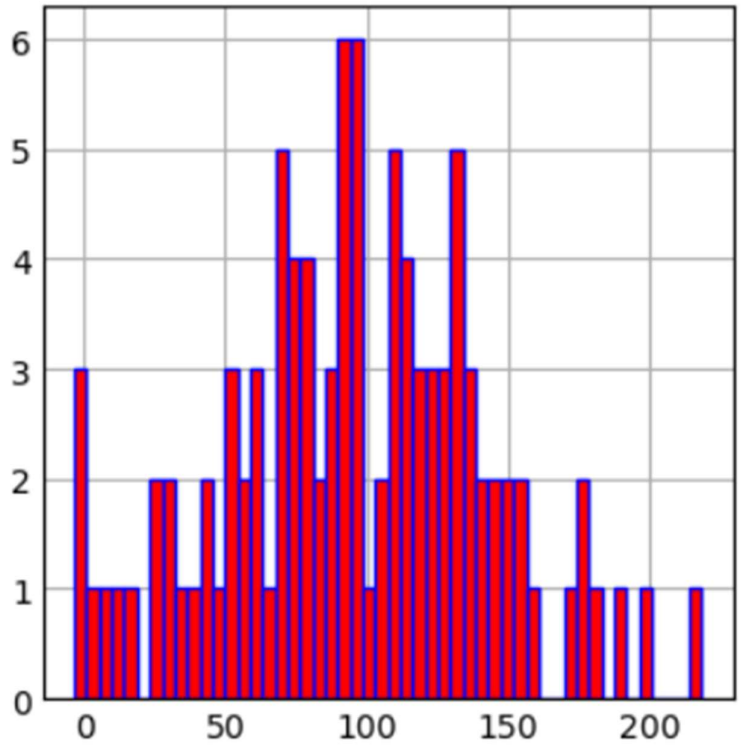
Histogram is a type of bar graph where the graph is represented in groups. hist() is used for plotting histogram with parameters x, bins, color, edgecolor.

Code:

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

x=np.random.normal(100,50,100)
fig = plt.figure(figsize = (3, 3))
plt.hist(x,bins=50,color="red", edgecolor="blue")
plt.show()
```

Output:



Description:

hist() have parameters x which is a random generated array around one hundred with standard deviation 50 of 100 values.

bins indicate the number of sections on x axis.

color indicates the color of rectangular bars.

edge color that divides the rectangular bars.

SCATTER PLOT

Scatter plot uses dots to depict the relationship between the data. scatter() is used for plotting scatter plot and scatter plot can be done for multiple datasets at the same time by differentiating it with colors. Legends help to achieve this difference.

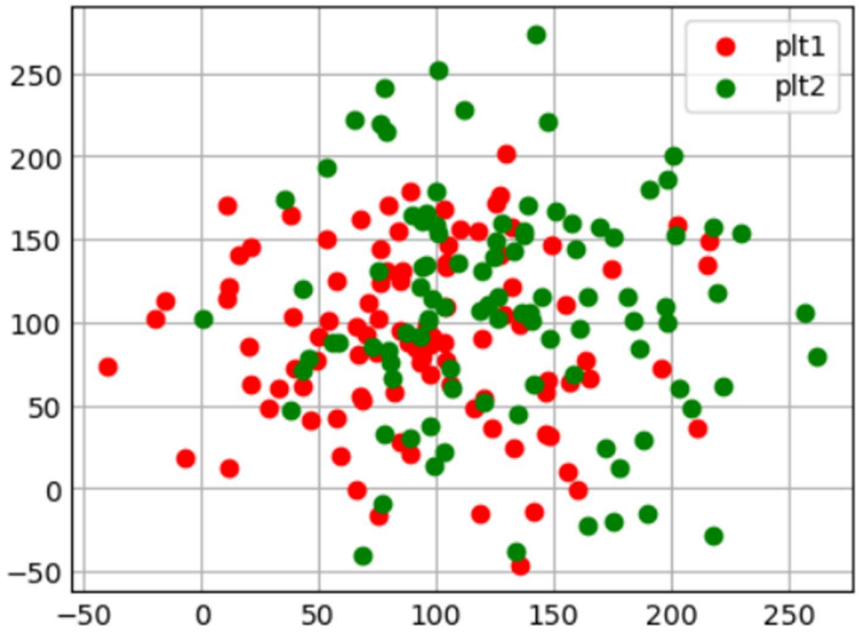
Code:

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

x1=np.random.normal(100,50,100)
y1=np.random.normal(100,50,100)
x2=np.random.normal(120,60,100)
y2=np.random.normal(120,60,100)
fig = plt.figure(figsize = (4, 3))
plt.scatter(x1,y1,c="r")
plt.scatter(x2,y2,c="g")
plt.legend(["plt1","plt2"])

plt.show()
```

Output:



Description:

x1, y1 belong to one dataset and x2,y2 belong to another dataset.

Here the scatter() is used for the scatter plot and the parameters x, y and c which represents color.

Legend adds the label which helps to differentiate the plots.

PIECHART

Pie charts are used to plot data of same kind which means the same series of data where the different elements are divide based on their percentage.

Code:

```
from matplotlib import pyplot as plt
import numpy as np
time = [8,8,4,4]
color=["r","g","y","b"]
work= ["sleeping","studying","dancing","others"]
fig = plt.figure(figsize=(4, 4))
plt.pie(time, labels=work, wedgeprops={"edgecolor":"black","linewidth":1}, colors=color)
plt.show()
```

Output:

Advantages of Matplotlib

- High customization and control over plots.
- Supports various types of plots.
- Strong community support.
- Suitable for publication-quality figures.

COMPARISON OF PANDAS AND MATPLOTLIB

Pandas and Matplotlib serve different but complementary purposes:

Pandas:
- Primarily used for data manipulation and analysis.
- Provides powerful data structures like Series and DataFrame.
- Can generate simple plots directly using `.plot()` function.

Matplotlib:
- Focused on data visualization with detailed customization.
- Provides low-level functions to create a variety of static and interactive plots.
- Often used with Pandas for plotting data stored in DataFrames.