np.random.randint(low, high=None, size=None, dtype=int); low(inclusive) & high(exclusive) or np.random.randint(high)
random.randint(a, b); low(inclusive) & high(inclusive)

Pandas: https://www.w3schools.com/python/pandas/default.asp
Matplotlib: https://www.w3schools.com/python/matplotlib_intro.asp

1. **What is the difference between a Pandas Series and a DataFrame?**
   - A **Pandas Series is a one-dimensional array**-like object that can hold any data type (integers, strings, floats, etc.). It has an associated index that labels each value.
   - A **Pandas DataFrame is a two-dimensional table** (like a spreadsheet or SQL table) with rows and columns. Each column can hold data of different types, and it has both row and column indices.

Task 1:

```
import pandas as pd
s=pd.Series(range(1,11))
print(s)
even=s[s%2==0]
odd=s[s%2!=0]
df=pd.DataFrame({'A': even.values, 'B': odd.values})
print(df)
```

1. Create a Pandas Series containing the numbers from 1 to 10.
2. Convert the Series into a DataFrame with two columns: "A" and "B". The first column should contain even numbers, and the second should contain odd numbers from 1 to 9.
3. Write some real life area where we can use series and Dataframe.

2. **What is the difference between .loc[] and .iloc[] for indexing and selecting data in a DataFrame?**
   - **.loc[]**: Label-based indexing. You specify the row and column labels.
   - **.iloc[]**: Integer-location based indexing. You specify the row and column positions (index positions, not labels).

```
# Using label-based indexing with .loc[]
df.loc[0, 'Name']  # Selects the value from the first row and 'Name' column

# Using integer-based indexing with .iloc[]
df.iloc[0, 1]  # Selects the value from the first row and second column
```

3. **What is the difference between merge() and join() functions in Pandas?**

   - merge(): Merges two DataFrames based on one or more common columns, like SQL joins (inner, outer, left, right).
   - join(): Joins two DataFrames based on their indexes or a common column. It's typically used for joining along the index.

Task 2:

1. Create two DataFrames. The first DataFrame should contain the columns 'ID', 'Name', and 'Age'. The second DataFrame should contain the columns 'ID' and 'Country'. Merge them on the 'ID' column.
2. Demonstrate the join() function by merging the two DataFrames from the previous question using a left join.

4. **How do you use the groupby() function in Pandas? Give an example of grouping by a column and calculating the mean of another column.**
5. **What is a Pivot Table in Pandas? How does it differ from a simple groupby operation?**

```python
import pandas as pd
s=pd.Series(range(1,11))
# print(s)
even=s[s%2==0]
odd=s[s%2!=0]
df=pd.DataFrame({'A': even.values, 'B': odd.values})
df2=pd.DataFrame({'A': even.values, 'A2': (even.values*10), 'B2':
(odd.values*10)})
print(df,"\n")
print(df2,"\n")
print(df.merge(df2),"\n") # foreign key concept (at least 1 column has to
have common name)
df2=pd.DataFrame({'A2': (even.values*10), 'B2': (odd.values*10)})
print(df.join(df2)) #combinig columns (column names must be unique)
```

```
    A  B
0   2  1
1   4  3
2   6  5
3   8  7
4  10  9

    A   A2  B2
0   2   20  10
1   4   40  30
2   6   60  50
3   8   80  70
4  10  100  90

    A  B   A2  B2
0   2  1   20  10
1   4  3   40  30
2   6  5   60  50
3   8  7   80  70
4  10  9  100  90

    A  B   A2  B2
0   2  1   20  10
1   4  3   40  30
2   6  5   60  50
3   8  7   80  70
4  10  9  100  90
```

Google Colab: https://colab.research.google.com/drive/1N8NrG1E-ZS70eSH03K5Dc5-Ewb3Vu-Hu#scrollTo=wOkWruaU0Zqa

Matplotlib is a powerful and flexible plotting library used in Python for creating static, animated, and interactive visualizations. It is widely used for data visualization because it can generate high-quality graphs and plots with minimal code. Whether you're plotting simple line graphs, scatter plots, or complex 3D visualizations, Matplotlib provides a wide range of features to customize your plots.

Key Features of Matplotlib:

1. **Wide Range of Plot Types**:
   - Line plots, bar plots, scatter plots, histograms, pie charts, heatmaps, box plots, and many more.
   - 2D and 3D plotting capabilities.
2. **Customization**:
   - Matplotlib allows for detailed customization of every element of the plot, from axes labels, colors, and tick marks to legends and annotations.
3. **Interactive Plots**:
   - Plots can be made interactive using Matplotlib's integration with IPython notebooks, Jupyter notebooks, and other tools.
4. **High-Quality Output**:
   - Matplotlib generates high-quality plots suitable for scientific publications, reports, or web apps.
5. **Integration with Other Libraries**:
   - Matplotlib can be integrated with libraries like **NumPy** (for numerical data manipulation) and **Pandas** (for data handling), making it a popular choice for data scientists and engineers.

How to Use Matplotlib:

pip install matplotlib

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

Basic Plotting with Matplotlib

1. Line Plot

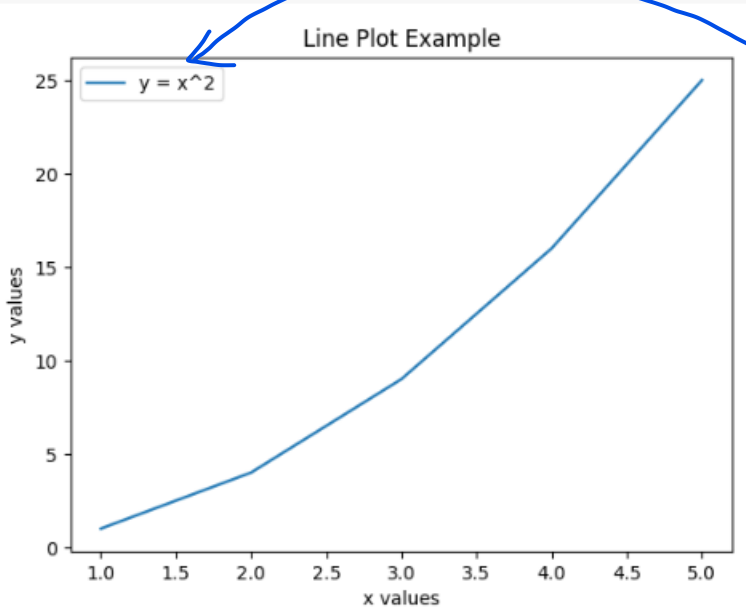
A line plot is one of the most common types of plots.

Code:

```
import matplotlib.pyplot as plt
# Data
x = [1, 2, 3, 4, 5]
y = [1, 4, 9, 16, 25]
# Plot
plt.plot(x, y, label='y = x^2')
# Adding title and labels
plt.title('Line Plot Example')
plt.xlabel('x values')
```

plt.ylabel('y values')
# Display the plot
plt.legend()
plt.show()

```
import matplotlib.pyplot as plt

# Data
x = [1, 2, 3, 4, 5]
y = [1, 4, 9, 16, 25]

# Plot
plt.plot(x, y, label='y = x^2')

# Adding title and labels
plt.title('Line Plot Example')
plt.xlabel('x values')
plt.ylabel('y values')

# Display the plot
plt.legend()
plt.show()
```



Ø if no legend

Remove legend() and analyze the output.

Line Plot with annotation: Write the following code and check the output

```
import matplotlib.pyplot as plt
# Data
x = [1, 2, 3, 4, 5]
y = [1, 4, 9, 16, 25]
# Plot with label for the legend
plt.plot(x, y, label='y = x^2')
# Annotate a point
plt.annotate('Point (3, 9)', xy=(3, 9), xytext=(4, 10),
             arrowprops=dict(arrowstyle="->", color='blue'))
```

```
# Adding title, labels, and legend
plt.title('Line Plot with Annotation')
plt.xlabel('x values')
plt.ylabel('y values')
plt.legend()
# Display the plot
plt.show()
```

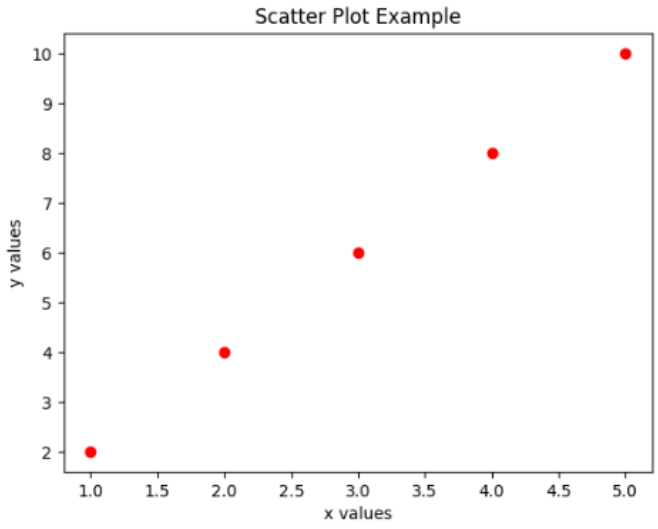
2. Scatter Plot

A scatter plot is useful for visualizing the relationship between two numerical variables.

```
import matplotlib.pyplot as plt
# Data
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]
# Scatter plot
plt.scatter(x, y, color='red')
# Adding title and labels
plt.title('Scatter Plot Example')
plt.xlabel('x values')
plt.ylabel('y values')
```

```
import matplotlib.pyplot as plt

# Data
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

# Scatter plot
plt.scatter(x, y, color='red')

# Adding title and labels
plt.title('Scatter Plot Example')
plt.xlabel('x values')
plt.ylabel('y values')
```

```
Text(0, 0.5, 'y values')
```



3. Bar Plot

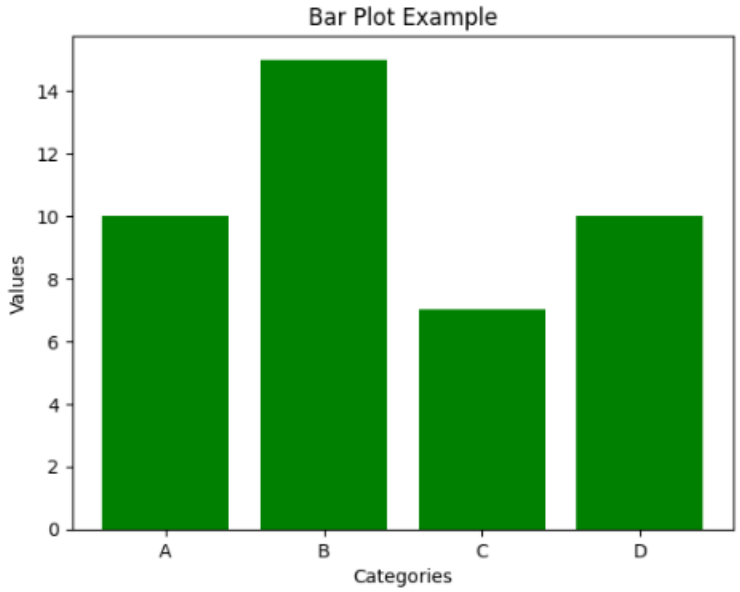
Bar plots are useful for visualizing categorical data.

import matplotlib.pyplot as plt

# Data
categories = ['A', 'B', 'C', 'D']
values = [10, 15, 7, 10]
# Bar plot
plt.bar(categories, values, color='green')
# Adding title and labels
plt.title('Bar Plot Example')
plt.xlabel('Categories')
plt.ylabel('Values')

```
import matplotlib.pyplot as plt

# Data
categories = ['A', 'B', 'C', 'D']
values = [10, 15, 7, 10]

# Bar plot
plt.bar(categories, values, color='green')

# Adding title and labels
plt.title('Bar Plot Example')
plt.xlabel('Categories')
plt.ylabel('Values')
```

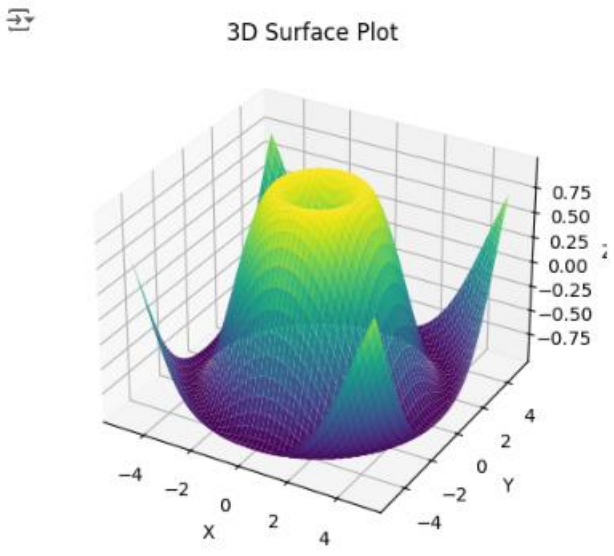
```
Text(0, 0.5, 'Values')
```



4. 3D Plotting
Matplotlib also supports 3D plotting. You can create 3D plots using Axes3D.

```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import numpy as np

# Data for a 3D plot
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
x, y = np.meshgrid(x, y)
z = np.sin(np.sqrt(x**2 + y**2))

# Create a 3D surface plot
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, z, cmap='viridis')

# Adding title and labels
ax.set_title('3D Surface Plot')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')

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

```python
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import numpy as np

# Data for a 3D plot
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
x, y = np.meshgrid(x, y)
z = np.sin(np.sqrt(x**2 + y**2))

# Create a 3D surface plot
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, z, cmap='viridis')

# Adding title and labels
ax.set_title('3D Surface Plot')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')

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



Subplots

Matplotlib allows you to create multiple plots in a single figure using subplots. This is useful when you want to compare different visualizations side by side.

Let's examine by a Titanic Dataset

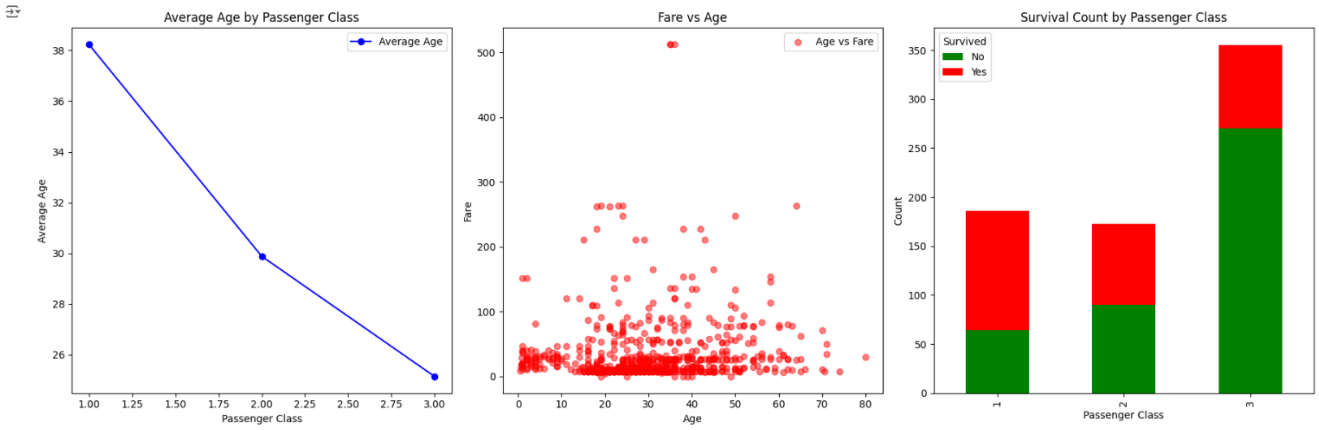
The **Titanic dataset** is a well-known dataset containing information about passengers on the Titanic, such as their age, gender, fare, and whether they survived. You can load the Titanic dataset directly from the seaborn library, which provides an easy-to-use interface for loading many popular datasets, including the Titanic dataset.

Steps:

1. Load the Titanic dataset using **Pandas**.
2. Create a line plot showing the average age of passengers over different passenger classes.
3. Create a scatter plot comparing the fare of passengers versus their age.
4. Create a bar plot to compare the number of survivors (Survived column) by passenger class (Pclass).

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Load Titanic dataset
titanic = sns.load_dataset('titanic')

# Drop rows with missing values for simplicity (you can also choose to fill missing values)
titanic.dropna(subset=['age', 'fare', 'pclass'], inplace=True)

# Create subplots (1 row, 3 columns)
fig, axs = plt.subplots(1, 3, figsize=(18, 6))

# **First Plot: Line Plot (Average Age by Passenger Class)**
avg_age_by_class = titanic.groupby('pclass')['age'].mean()
axs[0].plot(avg_age_by_class.index, avg_age_by_class.values, marker='o', color='b', label='Average Age')
axs[0].set_title('Average Age by Passenger Class')
axs[0].set_xlabel('Passenger Class')
axs[0].set_ylabel('Average Age')
axs[0].legend()

# **Second Plot: Scatter Plot (Fare vs Age)**
axs[1].scatter(titanic['age'], titanic['fare'], alpha=0.5, color='r', label='Age vs Fare')
axs[1].set_title('Fare vs Age')
axs[1].set_xlabel('Age')
axs[1].set_ylabel('Fare')
axs[1].legend()

# **Third Plot: Bar Plot (Survival Count by Passenger Class)**
survival_count_by_class = titanic.groupby('pclass')['survived'].value_counts().unstack().fillna(0)
survival_count_by_class.plot(kind='bar', stacked=True, color=['green', 'red'], ax=axs[2])
axs[2].set_title('Survival Count by Passenger Class')
axs[2].set_xlabel('Passenger Class')
axs[2].set_ylabel('Count')
axs[2].legend(title='Survived', labels=['No', 'Yes'])
```

```
# Adjust layout to avoid overlapping
plt.tight_layout()

# Show the plots
plt.show()
```



What is Seaborn?