1. Write a Python program using NumPy to find the mean and standard deviation of a list of numbers.

```
import numpy as np

data = [10, 20, 30, 40, 50]
mean = np.mean(data)
std_dev = np.std(data)

print("Mean:", mean)
print("Standard Deviation:", std_dev)
```

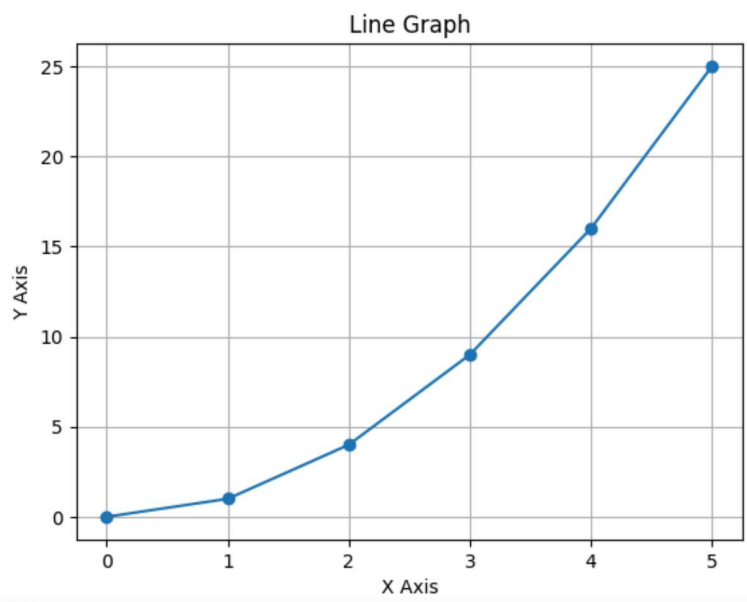
```
Mean: 30.0
Standard Deviation: 14.142135623730951
```

2. Write a Python program to plot a line graph using NumPy arrays (no random numbers).

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

x = np.array([0, 1, 2, 3, 4, 5])
y = np.array([0, 1, 4, 9, 16, 25])

plt.plot(x, y, marker='o')
plt.title('Line Graph')
plt.xlabel('X Axis')
plt.ylabel('Y Axis')
plt.grid(True)
plt.show()
```

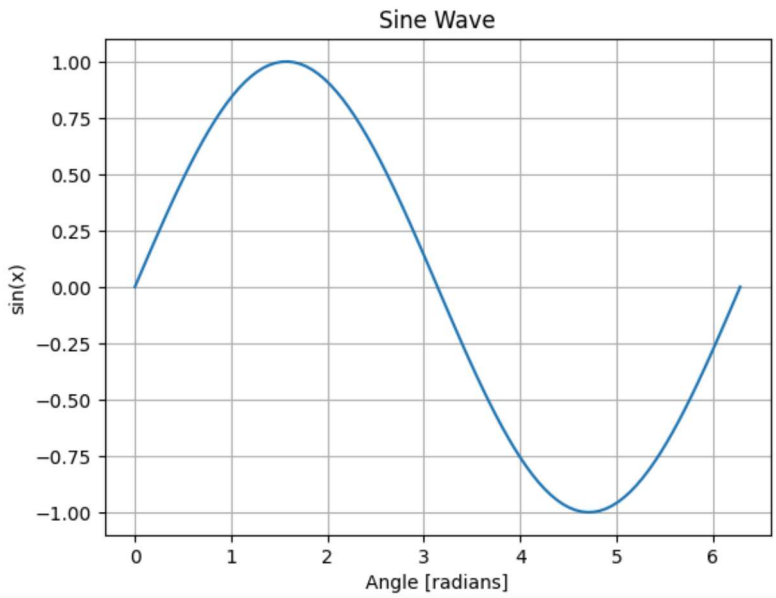


3. Write a Python program to plot a sine wave using NumPy and Matplotlib.

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

x = np.linspace(0, 2 * np.pi, 100)
y = np.sin(x)

plt.plot(x, y)
plt.title("Sine Wave")
plt.xlabel("Angle [radians]")
plt.ylabel("sin(x)")
plt.grid(True)
plt.show()
```



4. Linear Regression on House Prices

- **Task:** Predict house prices using area as a feature (1D Linear Regression).
- **Visualize**: Plot the data points and the best-fit regression line.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Area in square feet
area = np.array([500, 750, 1000, 1250, 1500]).reshape(-1, 1)
# Price in lakhs
price = np.array([50, 65, 80, 100, 120])

model = LinearRegression()
model.fit(area, price)

predicted_price = model.predict(area)

plt.scatter(area, price, color='blue', label='Actual')
plt.plot(area, predicted_price, color='red', label='Regression Line')
plt.title("Linear Regression - House Prices")
plt.xlabel("Area (sq ft)")
plt.ylabel("Price (lakhs)")
plt.legend()
plt.grid(True)
plt.show()
```

5. Logistic Regression for Binary Classification

- **Task:** Classify points into two categories using logistic regression (e.g., pass/fail based on hours studied).
- **Visualize:** Plot data points and decision boundary.

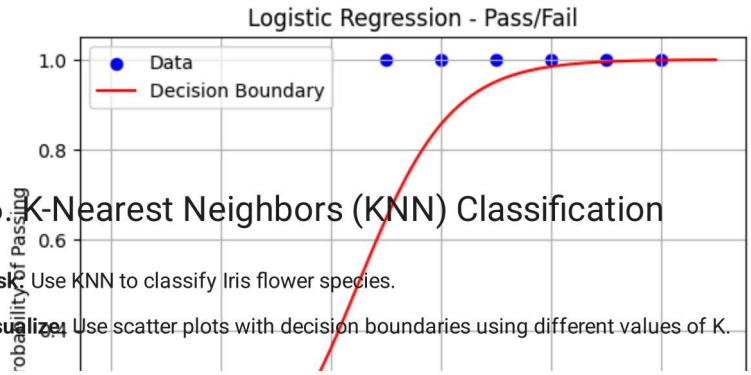
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression

# Hours studied
hours = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]).reshape(-1, 1)
# Pass (1) or Fail (0)
result = np.array([0, 0, 0, 0, 1, 1, 1, 1, 1, 1])

model = LogisticRegression()
model.fit(hours, result)

x_test = np.linspace(0, 11, 100).reshape(-1, 1)
y_prob = model.predict_proba(x_test)[:, 1]

plt.scatter(hours, result, color='blue', label='Data')
plt.plot(x_test, y_prob, color='red', label='Decision Boundary')
plt.title("Logistic Regression - Pass/Fail")
plt.xlabel("Hours Studied")
plt.ylabel("Probability of Passing")
plt.grid(True)
plt.legend()
plt.show()
```



6. K-Nearest Neighbors (KNN) Classification

a. **Task:** Use KNN to classify Iris flower species.

b. Visualize: Use scatter plots with decision boundaries using different values of K.

```
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.neighbors import KNeighborsClassifier

# Load iris dataset
iris = load_iris()
X = iris.data[:, :2]  # Use only first 2 features (sepal length and width)
y = iris.target

# Train KNN model (k=3)
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X, y)

# Plot data
for i, color in zip([0, 1, 2], ['red', 'green', 'blue']):
    plt.scatter(X[y == i, 0], X[y == i, 1], color=color, label=iris.target_names[i])

plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.title('KNN Classification (k=3)')
plt.legend()
plt.show()
```