write a Python program to calculate the Euclidean distance of a vector from the origin

```python
import numpy as np

vector = [2,3,4,5,6,7]



def distance_from_origin(vector):
    return np.linalg.norm(vector) # (L2 norm)

dist = distance_from_origin(vector)

print(f"The distance from the origin is: {dist:.4f}")
```

[11] ✓ 0.1s Python

```
The distance from the origin is: 11.7898
```

Write a Python program to calculate the Euclidean distance between two points

```python
def euclidean_distance(point1,point2):
    # converting to numpy array
    p1 = np.array(point1)
    p2 = np.array(point2)

    # calculating distance
    diff = p1 - p2

    return np.linalg.norm(diff)


dist = euclidean_distance([1,2,3,4,5],[7,5,3,2,1])

print(f"Euclidean distance: {dist:.4f}")
```

[12] ✓ 0.0s Python

```
Euclidean distance: 8.0623
```

Write a Python function to mean center a given 2-dimensional vector

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

def mean_center(vectors):
    # Calculating the mean for each dimension
    mean = np.mean(vectors, axis=0)

    # Subtract the mean from each vector
    centered_vector = vectors - mean
    return centered_vector

# Generating random 2D data with 100 points
data = np.random.rand(100, 2)

# Mean centering the data
center_data = mean_center(data)
```

[19] ✓ 0.0s Python

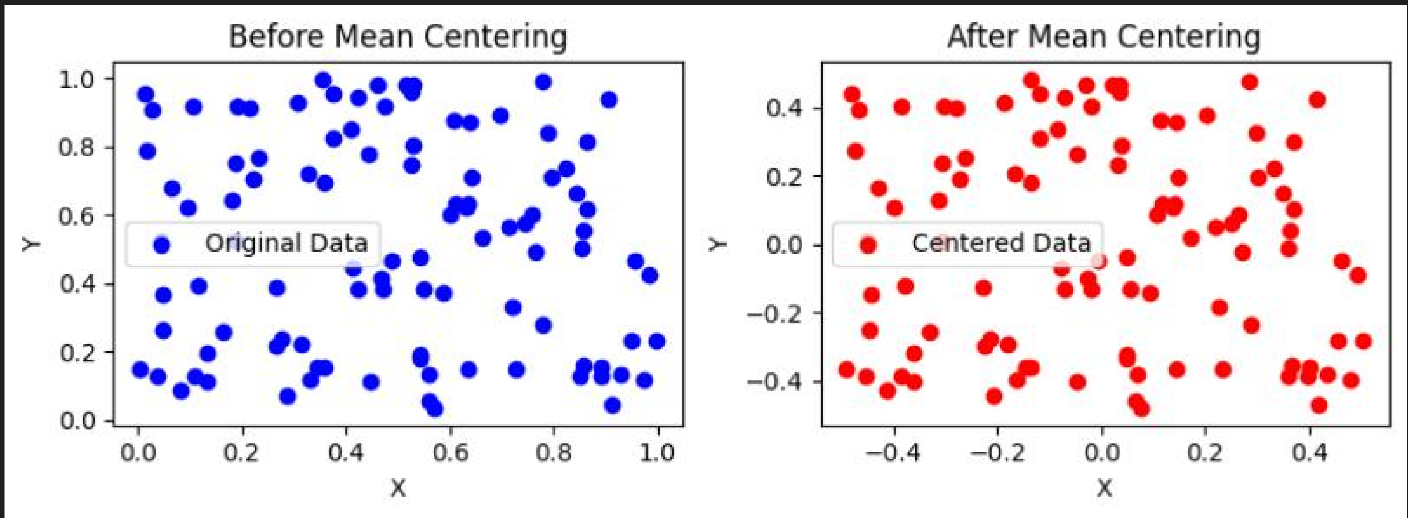
```python
# Create a figure and axis for plotting
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(8, 3))

# Scatter plot before mean centering
ax1.scatter(data[:, 0], data[:, 1], color='blue', label='Original Data')
ax1.set_title('Before Mean Centering')
ax1.set_xlabel('X')
ax1.set_ylabel('Y')
ax1.legend()

# Scatter plot after mean centering
ax2.scatter(center_data[:, 0], center_data[:, 1], color='red', label='Centered Data')
ax2.set_title('After Mean Centering')
ax2.set_xlabel('X')
ax2.set_ylabel('Y')
ax2.legend()

# Show the plot
plt.tight_layout()
plt.show()
```

✓ 0.1s Python



write a Python code to compute the dot product between two vectors using NumPy

```python
import numpy as np

# Example vectors
vector1 = np.array([1, 2, 3])
vector2 = np.array([4, 5, 6])

dot_product = np.dot(vector1,vector2)

print(f"The dot product of the two vectors is: {dot_product}")
```

[21] ✓ 0.0s Python

```
The dot product of the two vectors is: 32
```

write a code to check the commutative property vector_A dot vector_B == vector_B dot vecto_A

```python
# Example vectors
vectorA = np.array([1, 2, 3])
vectorB = np.array([4, 5, 6])

dot_product_of_v_AB = np.dot(vectorA,vectorB)
dot_product_of_v_BA = np.dot(vectorB,vectorA)

if dot_product_of_v_AB == dot_product_of_v_BA:
    print('The dot product is commutative')
else:
    print('The dot product is not commutative')
```

[22] ✓ 0.0s Python

```
The dot product is commutative
```

write a python code to check the distributed property of a vectors

```python
import numpy as np

def check_distributive_property(a,b,c):
    left_hand_side = np.dot(a,b+c)
    right_hand_side = np.dot(a,b) + np.dot(a,c)

    # check if both side are equal:
    return np.allclose(left_hand_side,right_hand_side)

# Example vectors
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
c = np.array([7, 8, 9])

# Check the distributive property
result = check_distributive_property(a, b, c)

if result:
    print("The distributive property holds.")
else:
    print("The distributive property does not hold.")
```

23] ✓ 0.0s Python

```
The distributive property holds.
```

write a python code to find the similarity between vectors

```python
import numpy as np

def cosine_similarity(vector1,vector2):

    dot_product = np.dot(vector1,vector2)

    norm_v1 = np.linalg.norm(vector1) # magnitude of v1
    norm_v2 = np.linalg.norm(vector2) # magnitude of v2

    # cosine simiarity
    similarity = dot_product / (norm_v1 * norm_v2)
    return similarity

# Example vectors
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])

# Calculate the cosine similarity
similarity_score = cosine_similarity(a, b)

print(f"The cosine similarity between the vectors is: {similarity_score:.4f}")
```

[24] ✓ 0.0s Python

```
The cosine similarity between the vectors is: 0.9746
```

write a python code to find the determinant of a matrix

```python
import numpy as np

def determinant(matrix):
    return np.linalg.det(matrix)

# Example matrix
matrix = np.array([[2, 4], [2, 4]])

# Find the determinant
det_result = determinant(matrix)

print(f"The determinant of the matrix is: {det_result}")
```

[29] ✓ 0.0s Python

```
The determinant of the matrix is: 0.0
```

write a python code to calculate the inverse of a matrix

```python
import numpy as np

def matrix_inverse(matrix):
    # Calculate the inverse using numpy's linalg.inv function
    try:
        return np.linalg.inv(matrix)
    except np.linalg.LinAlgError:
        return "Matrix is singular and cannot be inverted."

# Example matrix
matrix = np.array([[1, 2], [3, 4]])

# Find the inverse
inverse_matrix = matrix_inverse(matrix)

print("Original Matrix:")
print(matrix)

print("\nInverse of the Matrix:")
print(inverse_matrix)
```

[30] ✓ 0.0s Python

```
Original Matrix:
[[1 2]
 [3 4]]

Inverse of the Matrix:
[[-2.   1. ]
 [ 1.5 -0.5]]
```