Lab 1 solution key

1.

```
import numpy as np

# Define two vectors vector a and vector b
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
alpha = 2.5  # Input scalar value

# 1. Compute vector addition
vec_add = a + b

# 2. Compute scalar multiplication
scalar_mul_a = alpha * a
scalar_mul_b = alpha * b

# 3. Find the length (norm) of the vectors
len_a = np.linalg.norm(a)
len_b = np.linalg.norm(b)

# 4. Compute the dot product
dot_prod = np.dot(a, b)

# 5. Compute the cross product
cross_prod = np.cross(a, b)

# Output results
print("Vector Addition (a + b):", vec_add)
print(f"Scalar Multiplication ({alpha} * a):", scalar_mul_a)
print(f"Scalar Multiplication ({alpha} * b):", scalar_mul_b)
print("Length of Vector a:", len_a)
print("Length of Vector b:", len_b)
print("Dot Product (a . b):", dot_prod)
print("Cross Product (a x b):", cross_prod)
```

2.

```
import numpy as np
# Define two m x n matrices matrix A and matrix B
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])
```

```
# 1. Compute the matrix addition
mat_add = A + B

# 2. Compute matrix multiplication
if A.shape[1] == B.shape[0]:
    mat_mul = np.matmul(A, B)
else:
    mat_mul = "Invalid input"

# 3. Compute the transpose of both matrices
trans_A = A.T
trans_B = B.T

# 4. Compute the rank and nullity of matrix A and matrix B
rank_A = np.linalg.matrix_rank(A)
nullity_A = A.shape[1] - rank_A

rank_B = np.linalg.matrix_rank(B)
nullity_B = B.shape[1] - rank_B

# Output results
print("Matrix Addition (A + B):\n", mat_add)
print("Matrix Multiplication (A * B):\n", mat_mul)
print("Transpose of Matrix A:\n", trans_A)
print("Transpose of Matrix B:\n", trans_B)
print(f"Matrix A -> Rank: {rank_A}, Nullity: {nullity_A}")
print(f"Matrix B -> Rank: {rank_B}, Nullity: {nullity_B}")
```

3.

```
import numpy as np
# Define an n x n matrix A
A = np.array([[4, 7], [2, 6]])

# 1. Compute the determinant of the matrix
det_A = np.linalg.det(A)

# 2. Compute the inverse of A
if np.isclose(det_A, 0):
    inv_A = "The matrix is not invertible"
else:
    inv_A = np.linalg.inv(A)

# 3. Compute eigen values and eigen vectors of A
```

```
eigenvalues, eigenvectors = np.linalg.eig(A)

# Output results
print("Determinant:", det_A)
print("Inverse of Matrix A:\n", inv_A)
print("Eigenvalues:", eigenvalues)
print("Eigenvectors:\n", eigenvectors)
```

4

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

# Define two arrays x and y
x = np.array([1, 2, 3, 4, 5])
y = np.array([10, 24, 36, 40, 52])

# Plot x over y using line chart, bar chart, histogram, scatter plot, box plot, pie chart
fig, axes = plt.subplots(2, 3, figsize=(14, 9))

# Line Chart
axes[0, 0].plot(x, y, marker="o", color="blue")
axes[0, 0].set_title("Line Chart")
axes[0, 0].set_xlabel("x")
axes[0, 0].set_ylabel("y")

# Bar Chart
axes[0, 1].bar(x, y, color="skyblue")
axes[0, 1].set_title("Bar Chart")
axes[0, 1].set_xlabel("x")
axes[0, 1].set_ylabel("y")

# Histogram
axes[0, 2].hist(y, bins=5, color="green", edgecolor="black")
axes[0, 2].set_title("Histogram")
axes[0, 2].set_xlabel("y values")

# Scatter Plot with two random datasets
np.random.seed(42)
rand_data1 = np.random.randn(50)
rand_data2 = np.random.randn(50)
axes[1, 0].scatter(rand_data1, rand_data2, color="purple")
axes[1, 0].set_title("Scatter Plot (2 Random Datasets)")
axes[1, 0].set_xlabel("Dataset 1")
axes[1, 0].set_ylabel("Dataset 2")
```

```
# Box Plot
axes[1, 1].boxplot(y)
axes[1, 1].set_title("Box Plot")
axes[1, 1].set_ylabel("y values")

# Pie Chart
axes[1, 2].pie(y, labels=x, autopct="%1.1f%%")
axes[1, 2].set_title("Pie Chart")

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