To demonstrate solid programming skills using libraries like NumPy and Matplotlib, you can showcase your proficiency in various tasks such as data manipulation, analysis, visualization, and numerical computing. Here's a demonstration of programming skills using these libraries:

Data Manipulation with NumPy:

Create NumPy arrays, manipulate array elements, and perform basic arithmetic operations.

Utilize slicing, indexing, and broadcasting to manipulate arrays efficiently.

Demonstrate knowledge of array reshaping, stacking, and splitting.

Example:

python

Copy code

import numpy as np

# Create a NumPy array

arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

# Perform array manipulation

arr\_sum = np.sum(arr)

arr\_mean = np.mean(arr, axis=0)

arr\_transpose = np.transpose(arr)

Data Visualization with Matplotlib:

Create various types of plots such as line plots, scatter plots, histograms, bar plots, and pie charts.

Customize plot aesthetics including colors, markers, line styles, labels, and annotations.

Combine multiple plots into a single figure, and adjust subplot arrangements.

Example:

python

Copy code

import matplotlib.pyplot as plt

# Generate data

x = np.linspace(0, 10, 100)

y1 = np.sin(x)

y2 = np.cos(x)

# Create a figure with subplots

fig, ax = plt.subplots(2, 1, figsize=(8, 6))

# Plot data on each subplot

ax[0].plot(x, y1, color='blue', label='sin(x)')

ax[1].plot(x, y2, color='red', label='cos(x)')

# Add titles and legends

ax[0].set\_title('Sine Wave')

ax[1].set\_title('Cosine Wave')

ax[0].legend()

ax[1].legend()

# Show plot

plt.tight\_layout()

plt.show()

Numerical Computing with NumPy:

Perform matrix operations, linear algebra calculations, and statistical computations.

Utilize NumPy functions for mathematical operations, including trigonometric, exponential, and logarithmic functions.

Demonstrate knowledge of random number generation and statistical distributions.

Example:

python

Copy code

import numpy as np

# Generate random data

data = np.random.randn(1000)

# Calculate mean and standard deviation

data\_mean = np.mean(data)

data\_std = np.std(data)

# Generate histogram

plt.hist(data, bins=30, density=True, alpha=0.5, color='blue')

# Plot Gaussian distribution

x = np.linspace(data.min(), data.max(), 100)

y = (1 / (data\_std \* np.sqrt(2 \* np.pi))) \* np.exp(-(x - data\_mean)\*\*2 / (2 \* data\_std\*\*2))

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

# Add labels and title

plt.xlabel('Value')

plt.ylabel('Frequency')

plt.title('Histogram with Gaussian Fit')

# Show plot

plt.show()

By demonstrating proficiency in these areas, you can showcase your solid programming skills using libraries like NumPy and Matplotlib for various data-related tasks.