30 NumPy Projects

Complete Project Instructions

Level 1: Beginner (Basics & Indexing)

Project 1: Stats Calculator

What to build: A program that calculates basic statistics from a list of numbers.

Instructions:

1. Create a NumPy array with exactly 20 numbers (you can choose any numbers)
2. Calculate and print the following:
    - Mean (average)
    - Median (middle value)
    - Minimum value
    - Standard deviation
3. Use NumPy functions: np.mean(), np.median(), np.min(), np.std()

Example Output:

Numbers: [12, 15, 18, 22, 25, ...]

Mean: 45.5

Median: 44.0

Minimum: 12

Standard Deviation: 15.3

Project 2: Pass/Fail Counter

What to build: A program that counts how many students passed an exam.

Instructions:

1. Create a NumPy array with 30 student marks (random numbers between 0 and 100)
2. Count how many students scored **more than 40** (passing grade)
3. Print the total number of students who passed
4. **Hint:** Use np.sum() with a condition like marks > 40

Example Output:

Total students: 30

Students who passed (>40): 23

Students who failed: 7

Project 3: Matrix Operations

What to build: Basic operations on two matrices.

Instructions:

1. Create two 3×3 matrices using np.array()
    - Example: matrix1 = np.array([[1,2,3], [4,5,6], [7,8,9]])
2. Calculate and print:
    - Sum of both matrices (add them together)
    - Difference (subtract matrix2 from matrix1)
    - Transpose of the first matrix (flip rows and columns)
3. Use +, -, and .T for transpose

Example Output:

Matrix 1:

[[1 2 3]

[4 5 6]

[7 8 9]]

Sum of matrices:

[[2 4 6]

[8 10 12]

[14 16 18]]

Project 4: Dice Simulator

What to build: Simulate rolling a dice 100 times and count how many sixes you get.

Instructions:

1. Generate 100 random numbers between 1 and 6 using np.random.randint(1, 7, 100)
2. Count how many times you rolled a 6
3. Print all the rolls and the count of sixes
4. **Hint:** Use np.sum(rolls == 6) to count

Example Output:

100 dice rolls: [3, 6, 2, 1, 6, 4, ...]

Number of times 6 appeared: 18

Project 5: Temperature Converter

What to build: Convert Celsius temperatures to Fahrenheit.

Instructions:

1. Create an array of 10 Celsius temperatures (e.g., [0, 10, 20, 30, ...])
2. Convert ALL temperatures to Fahrenheit using the formula: **F = (C × 9/5) + 32**
3. Print both arrays side by side
4. **Hint:** You can multiply/add to entire arrays at once!

Example Output:

Celsius: [0 10 20 30 40]

Fahrenheit: [32 50 68 86 104]

Project 6: Odd/Even Filter

What to build: Filter out only even numbers from a list.

Instructions:

1. Generate 50 random integers between 1 and 100 using np.random.randint()
2. Use **boolean masking** to select only even numbers
3. Print the original array and the filtered even numbers
4. **Hint:** even_numbers = numbers[numbers % 2 == 0]

Example Output:

Original (50 numbers): [23, 44, 67, 12, 88, ...]

Even numbers only: [44, 12, 88, 56, 92, ...]

Project 7: Random Distributions

What to build: Compare uniform and normal random distributions.

Instructions:

1. Generate two arrays of 1000 numbers each:
    - Array 1: Uniform distribution using np.random.uniform(0, 100, 1000)
    - Array 2: Normal distribution using np.random.normal(50, 15, 1000) (mean=50, std=15)
2. For each array, calculate and print:
    - Mean

- Maximum value

3. Observe which distribution has values closer to the mean

Example Output:

Uniform Distribution - Mean: 49.8, Max: 99.2

Normal Distribution - Mean: 50.1, Max: 92.3

Project 8: Simple Grading

What to build: Automatically assign "Pass" or "Fail" to student scores.

Instructions:

1. Create an array of 20 student scores (0-100)
2. Use np.where() to create a new array with "Pass" where score > 50, else "Fail"
3. Print both arrays together
4. **Syntax:** results = np.where(scores > 50, "Pass", "Fail")

Example Output:

Scores: [45, 67, 52, 38, 71, ...]

Result: ['Fail', 'Pass', 'Pass', 'Fail', 'Pass', ...]

Project 9: Top 3 Ranker

What to build: Find the fastest three runners in a race.

Instructions:

1. Create an array of 10 race times in seconds (e.g., [12.5, 11.2, 13.8, ...])
2. Find the **indices** of the 3 fastest (smallest) times
3. Print the top 3 times and their positions
4. **Hint:** Use np.argsort()[:3] to get indices of smallest values

Example Output:

All race times: [12.5, 11.2, 13.8, 10.9, 14.2, ...]

Top 3 positions: [3, 1, 0]

Top 3 times: [10.9, 11.2, 12.5]

Project 10: Element-wise Calculator

What to build: Perform math operations between two arrays.

Instructions:

1. Create two arrays of 5 numbers each
2. Calculate and print:
    - Element-wise sum (add corresponding elements)
    - Element-wise product (multiply corresponding elements)
    - Square of differences: $(array1 - array2)^2$
3. All operations work element-by-element

Example Output:

Array 1: [2, 4, 6, 8, 10]

Array 2: [1, 3, 5, 7, 9]

Sum: [3, 7, 11, 15, 19]

Product: [2, 12, 30, 56, 90]

Squared difference: [1, 1, 1, 1, 1]

Level 2: Intermediate (2D Arrays & Data)

Project 11: Attendance Check

What to build: Track student attendance and calculate attendance percentages.

Instructions:

1. Create a 2D array (matrix) with shape (5 students × 10 days)
2. Fill with 1 (present) or 0 (absent) randomly using np.random.randint(0, 2, (5, 10))
3. For each student (each row), calculate attendance percentage
4. Print the attendance matrix and percentages
5. **Hint:** Sum across columns for each row using np.sum(matrix, axis=1)

Example Output:

Attendance Matrix (1=Present, 0=Absent):

[[1 0 1 1 1 0 1 1 1 0]

[1 1 1 0 1 1 1 1 0 1]

...]

Student 1: 70% attendance

Student 2: 80% attendance

...

Project 12: Expense Tracker

What to build: Analyze daily expenses for one month.

Instructions:

1. Create an array of 30 random expenses between $10 and $200
2. Calculate:
    - Total expense for the month
    - Average daily expense
    - Day with the highest expense (use np.argmax())
    - Total expense per week (split into 4 weeks, hint: reshape or slice)
3. Print all results clearly

Example Output:

Daily expenses: [45, 120, 78, 34, ...]

Total for month: $2,450

Average per day: $81.67

Highest expense on Day 15: $198

Week 1 total: $520

Project 13: Weather Analysis

What to build: Find the hottest and coldest days from temperature data.

Instructions:

1. Create an array of 30 daily temperatures (°C) between -5 and 40
2. Find:
    - Index of the hottest day using np.argmax()
    - Index of the coldest day using np.argmin()
    - Temperature on those days
3. Print the results with day numbers (add 1 to index for human-readable day)

Example Output:

Temperatures for 30 days: [22, 25, 18, ...]

Hottest day: Day 17 with 38°C

Coldest day: Day 5 with -3°C

Project 14: Image Brightness

What to build: Increase the brightness of a grayscale image.

Instructions:

1. Create a 10×10 array representing a grayscale image with values 0-255
    - Use np.random.randint(0, 256, (10, 10))
2. Add 50 to every pixel to increase brightness
3. Use np.clip() to ensure no value goes above 255
4. Print original and brightened images (or a small sample)
5. **Syntax:** brightened = np.clip(image + 50, 0, 255)

Example Output:

Original image (sample):

[[45 78 123]

 [200 150 90]

 [30 60 210]]

Brightened (+50):

[[95 128 173]

 [250 200 140]

 [80 110 255]]

Project 15: Sales Matrix

What to build: Analyze monthly sales data for multiple products.

Instructions:

1. Create a 2D array: 5 products × 12 months with random sales (100-1000)
2. Calculate:
    - Total revenue per month (sum each column)

- Month with highest total revenue
- Best-selling product overall (sum each row)

3. **Hint:** Use axis=0 for column sums, axis=1 for row sums

Example Output:

Sales Matrix (Products × Months):

[[450 678 890 ...]

[320 550 720 ...]

...]

Month with highest revenue: Month 7 with $4,250

Best-selling product: Product 3 with $8,900 total

Project 16: Manual Histogram

What to build: Count how many people fall into different age groups.

Instructions:

1. Create an array of 100 random ages between 0 and 100
2. Count how many people are in each group:
   - **0-18**: Children
   - **19-35**: Young Adults
   - **36-60**: Middle-aged
   - **60+**: Seniors
3. Use boolean conditions to count each group
4. Print the counts

Example Output:

Total people: 100

Children (0-18): 18

Young Adults (19-35): 32

Middle-aged (36-60): 35

Seniors (60+): 15

Project 17: Min-Max Scaler

What to build: Scale any dataset to values between 0 and 1.

Instructions:

1. Create an array of 20 random numbers (any range)
2. Apply Min-Max scaling using the formula:
    - **scaled = (value - min) / (max - min)**
3. Verify that the smallest value becomes 0 and largest becomes 1
4. Write this as a reusable function

Example Code Structure:

```
def min_max_scale(data):
  # Your code here
  return scaled_data
```

Example Output:

Original: [10, 45, 78, 23, 90, ...]

Scaled: [0.0, 0.388, 0.75, 0.144, 1.0, ...]

Project 18: Z-Score Normalizer

What to build: Standardize data using Z-score normalization.

Instructions:

1. Create an array of 30 random numbers
2. Calculate:
    - Mean of the array
    - Standard deviation
3. Apply Z-score formula to each element: **z = (x - mean) / std**
4. The normalized data should have mean ≈ 0 and std ≈ 1
5. Verify by calculating mean and std of the normalized array

Example Output:

Original: [23, 45, 67, 12, ...]

Mean: 45.2, Std: 18.3

Normalized: [-1.21, -0.01, 1.19, -1.81, ...]

New Mean: 0.0, New Std: 1.0

Project 19: Data Cleaner

What to build: Replace missing values (NaN) with the mean.

Instructions:

1. Create an array of 20 numbers
2. Manually insert some NaN (Not a Number) values using np.nan
    - Example: data[5] = np.nan
3. Calculate the mean of non-NaN values using np.nanmean()
4. Replace all NaN values with this mean using np.where() or np.nan_to_num()
5. Print before and after

Example Output:

Original: [12, 45, nan, 67, nan, 23, ...]

Mean of valid data: 36.75

Cleaned: [12, 45, 36.75, 67, 36.75, 23, ...]

Project 20: Row vs. Column Operations

What to build: Calculate statistics across rows and columns of a matrix.

Instructions:

1. Create a 5×5 matrix with random integers (1-100)
2. Calculate:
    - Sum of each row (5 values) using axis=1
    - Maximum value of each column (5 values) using axis=0
3. Print the matrix and both results clearly

Example Output:

Matrix:

[[12 45 67 23 89]

[34 56 78 12 45]

...]

Row sums: [236, 225, ...]

Column maximums: [89, 78, 92, 67, 95]

Level 3: Advanced (Vectorization & Systems)

Project 21: Speed Benchmark

What to build: Compare Python loop vs NumPy speed.

Instructions:

1. Create an array of 1 million random numbers
2. **Method 1:** Sum using a Python for loop
    - Measure time using import time and time.time()
3. **Method 2:** Sum using np.sum()
    - Measure time the same way
4. Compare the times and calculate how many times faster NumPy is

Example Code Structure:

```
import numpy as np
import time

data = np.random.rand(1000000)

# Method 1: Python loop
start = time.time()
total = 0
for num in data:
    total += num
loop_time = time.time() - start

# Method 2: NumPy
start = time.time()
np_total = np.sum(data)
numpy_time = time.time() - start

print(f"Loop time: {loop_time:.4f}s")
print(f"NumPy time: {numpy_time:.4f}s")
print(f"NumPy is {loop_time/numpy_time:.1f}x faster")
```

Project 22: Cosine Similarity

What to build: Calculate similarity between two vectors (0 = different, 1 = identical).

Instructions:

1. Create two arrays representing feature vectors (e.g., [1, 2, 3] and [2, 4, 6])
2. Calculate cosine similarity using:
    - **similarity = (A · B) / (||A|| × ||B||)**
    - Dot product: np.dot(A, B)
    - Magnitude: np.linalg.norm(A)
3. Result should be between 0 and 1

Example Output:

Vector A: [1, 2, 3]

Vector B: [2, 4, 6]

Cosine Similarity: 1.0 (identical direction)

Vector A: [1, 0, 0]

Vector B: [0, 1, 0]

Cosine Similarity: 0.0 (perpendicular)

Project 23: Distance Calculator

What to build: Calculate distance between two points.

Instructions:

1. Create two arrays representing (x, y) coordinates
    - Point 1: [3, 4]
    - Point 2: [7, 1]
2. Calculate Euclidean distance: $\mathbf{d = \sqrt{[(x_2-x_1)^2 + (y_2-y_1)^2]}}$
3. Use NumPy: np.sqrt(np.sum((point1 - point2)**2))
4. Test with multiple point pairs

Example Output:

Point 1: [3, 4]

Point 2: [7, 1]

Distance: 5.0

Project 24: Linear Regression

What to build: Find the best-fit line for data points.

Instructions:

1. Create sample data:
    - X values: [1, 2, 3, 4, 5]
    - Y values: [2, 4, 5, 4, 5]
2. Calculate slope (m) and intercept (b) using least squares:
    - **m = (mean(X×Y) - mean(X)×mean(Y)) / (mean(X^2) - mean(X)^2)**
    - **b = mean(Y) - m×mean(X)**
3. Print the equation: y = mx + b

Example Output:

X: [1, 2, 3, 4, 5]

Y: [2, 4, 5, 4, 5]

Best fit line: y = 0.6x + 2.2

Slope: 0.6

Intercept: 2.2

Project 25: Signal Smoother

What to build: Smooth noisy data using a moving average.

Instructions:

1. Create an array of 100 numbers with random noise
2. Implement a moving average filter:
    - For each point, replace it with the average of itself and its neighbors
    - Window size = 5 (current point + 2 before + 2 after)
3. Use np.convolve() or manual loops
4. Compare original vs smoothed data

Example Code Hint:

```
window_size = 5
smoothed = np.convolve(data, np.ones(window_size)/window_size, mode='valid')
```

Example Output:

Original data (first 10): [23, 67, 12, 89, 45, ...]

Smoothed data (first 10): [45.2, 47.4, 46.6, ...]

Project 26: Monte Carlo Pi Estimation

What to build: Estimate π by throwing random darts at a circle.

Instructions:

1. Generate 10,000 random (x, y) points between -1 and 1
2. Count how many points fall inside a circle of radius 1
    - Inside if: $x^2 + y^2 \leq 1$
3. Estimate π using: **$\pi \approx 4 \times$ (points inside) / (total points)**
4. Compare your estimate to the actual value (3.14159...)

Example Output:

Total points: 10,000

Points inside circle: 7,854

Estimated π: 3.1416

Actual π: 3.14159

Error: 0.001%

Project 27: Random Walk

What to build: Simulate a particle taking random steps.

Instructions:

1. Start at position 0
2. Generate 1,000 random steps: either +1 or -1
    - Use np.random.choice([-1, 1], 1000)
3. Calculate cumulative sum to get position at each step
4. Print final position and plot (optional)

Example Output:

Steps: [-1, 1, 1, -1, 1, ...]

Final position after 1,000 steps: -23

Max distance from start: 67

Project 28: Recommendation Engine

What to build: Match a user to their favorite movie using preferences.

Instructions:

1. Create a user preference vector: [Action, Comedy, Drama, Horror] = [5, 2, 4, 1]
2. Create a movie features matrix (4 movies × 4 features):
3. Movie 1: [5, 1, 3, 0] # Action movieMovie 2: [1, 5, 2, 0] # ComedyMovie 3: [3, 2, 5, 1] # DramaMovie 4: [2, 1, 1, 5] # Horror
4. Calculate match scores using dot product: scores = np.dot(movies, user_preferences)
5. Recommend the movie with the highest score

Example Output:

User preferences: [5, 2, 4, 1] (loves action, likes drama)

Match scores:

Movie 1: 42

Movie 2: 23

Movie 3: 39

Movie 4: 18

Recommendation: Movie 1 (Action movie)

Project 29: Correlation Tool

What to build: Manually calculate correlation between two datasets.

Instructions:

1. Create two arrays of 20 numbers each
2. Calculate Pearson correlation coefficient WITHOUT using np.corrcoef():
    - **r = covariance(X,Y) / (std(X) × std(Y))**
    - Covariance = mean((X - mean(X)) × (Y - mean(Y)))
3. Result should be between -1 (negative correlation) and +1 (positive correlation)
4. Verify your answer using np.corrcoef()

Example Output:

Array X: [1, 2, 3, 4, 5, ...]

Array Y: [2, 4, 6, 8, 10, ...]

Manual calculation: 1.0 (perfect positive correlation)

NumPy verification: 1.0 ✓

Project 30: Parallel Coin Flip

What to build: Simulate thousands of coin flips at once.

Instructions:

1. Generate 10,000 coin flips using np.random.randint(0, 2, 10000)
    - 0 = Tails, 1 = Heads
2. Calculate:
    - Probability of Heads (should be ≈ 0.5)
    - Longest streak of consecutive Heads
    - Longest streak of consecutive Tails
3. Run the simulation 5 times and compare results

Example Output:

Simulation 1:

Total flips: 10,000

Heads: 5,043 (50.43%)

Tails: 4,957 (49.57%)

Longest Heads streak: 8

Longest Tails streak: 9

Simulation 2:

...

📋 **How to Choose Your Project**

- **Complete beginner?** Start with Projects 1-5
- **Know basic Python?** Try Projects 6-15
- **Ready for a challenge?** Go for Projects 16-30
- **Want to impress?** Projects 21, 24, 26, or 28

Good luck with your NumPy journey!