NumPy Cheat Sheet

1. Setup & Configuration

```
import numpy as np

# Formatting: Avoid exponential notation (e.g., 1e-4), set precision
np.set_printoptions(precision=4, suppress=True)
```

2. Array Creation

From Python Lists

```
arr = np.array([1, 2, 3])            # 1D Array
mat = np.array([[1, 2], [3, 4]])     # 2D Array (Matrix)
# Note: Arrays must be homogeneous (same data type).
# If types mix (e.g. int + str), NumPy upcasts to the most complex type (usually
str).
```

Placeholders & Ranges

```
np.zeros(5)              # 1D array of zeros
np.zeros((3, 3))         # 2D array of zeros
np.ones(5)               # 1D array of ones
np.eye(3)                # Identity matrix (1s on diagonal, 0s elsewhere)

np.arange(10)            # Range: [0, 1, ..., 9] (Exclusive stop)
np.arange(10, 20, 2)     # Range with start, stop, step
np.linspace(1, 5, 10)    # 10 evenly spaced numbers between 1 and 5 (Inclusive)
```

Random Generation

```
np.random.seed(1)                # Reproducibility
np.random.randn(5)               # Std Normal Dist (mean=0, std=1)
np.random.randint(5, 10, (2, 3))  # Random Integers: low (incl), high (excl),
shape
```

3. Inspection & Attributes

Attribute	Description	Example
`.shape`	Dimensions (rows, cols)	`(2, 3)`
`.ndim`	Number of dimensions	`2`
`.size`	Total number of elements	`6`
`.dtype`	Data type of elements	`int64`, `<U9` (Unicode str)
`.itemsize`	Memory size of one element	`8` (bytes for int64)

4. Indexing & Slicing

Basic Indexing

```
arr[0]          # Get element at index 0
arr[0, 0]       # Get element at row 0, col 0 (Preferred over arr[0][0])
```

Slicing `[start:stop:step]`

```
arr[:, 0]       # All rows, column 0
arr[0, :]       # Row 0, all columns
arr[:, -1]      # Last column
arr[1:-1, 1:-1] # Crop borders (exclude first/last rows and cols)
arr[::-1]       # Reverse array
```

Fancy Indexing & Masks

```
arr[[0, 2]]     # Select indices 0 and 2 explicitly
mask = arr > 5  # Boolean mask (True where condition met)
arr[mask]       # Select elements where mask is True
```

5. Shapes & Manipulation

Reshaping *Rule: New shape product must equal total size.*

```
arr.reshape(4, 4)      # Change to 4x4
arr.reshape(-1, 1)     # -1 infers dimension automatically
```

Dimensions

```
np.expand_dims(arr, 1) # Add dimension at index 1 (Vector -> Matrix)
np.squeeze(arr)        # Remove dims of size 1 (Matrix -> Vector)
arr.T                  # Transpose (Swap rows/cols)
```

Combining Arrays

```
# Stacking (New Axis)
np.stack((a, b), axis=0)      # Stack along new row axis
np.stack((a, b), axis=1)      # Stack along new col axis

# Concatenation (Existing Axis)
np.concatenate((a, b), axis=0) # Join vertically (extend rows)
np.hstack((a, b))              # Horizontal stack (equiv to axis=1)
np.vstack((a, b))              # Vertical stack (equiv to axis=0)
```

6. Math & Statistics

Aggregations

```
arr.sum()           # Sum of all elements
arr.sum(axis=0)     # Sum down columns (vertical result)
arr.sum(axis=1)     # Sum across rows (horizontal result)

arr.mean()          # Average
arr.std()           # Standard Deviation
arr.var()           # Variance
arr.min(), arr.max()
```

Logic & Search

```
np.argmax(arr)                # Index of max value
np.argmin(arr)                # Index of min value
np.where(arr > 5)             # Indices where condition is True
np.where(arr > 5, 'Hi', 'Lo')   # If True 'Hi', else 'Lo'
```

Vectorization & Broadcasting *Operations apply element-wise automatically.*

```
arr + 10            # Add 10 to every element
arr1 + arr2         # Add element-wise
```

```
arr1 * arr2         # Multiply element-wise
```

---

7. Practical Recipes (From Notebook Challenges)

A. Min-Max Normalization Scale data between 0 and 1.

```
_min = data.min(axis=0)
_max = data.max(axis=0)
normalized = (data - _min) / (_max - _min)
```

B. Outlier Removal (Z-Score Logic) Remove values far from the mean (e.g.,).

```
mean = data.mean()
std = data.std()
mask = np.abs(data - mean) > (2 * std) # Create mask for outliers
data[mask] = mean                      # Replace outliers (e.g., with mean)
```

C. Border Manipulation Setting the edges of a 2D array to a specific value.

```
data[:, 0] = 1   # Left col
data[:, -1] = 1  # Right col
data[0, :] = 1   # Top row
data[-1, :] = 1  # Bottom row
```

D. Image/Channel Ops Working with shape `(H, W, Channels)`.

```
# Mute Green channel (assuming RGB format: 0=R, 1=G, 2=B)
img[:, :, 1] = 0

# Masking based on color thresholds
# e.g., Red > 200 AND Blue < 50
mask = (img[:, :, 0] > 200) & (img[:, :, 2] < 50)
```

E. Nearest Neighbor (Euclidean Distance) Find the point closest to origin (or another point). Formula:

```
# Using linalg
distances = np.linalg.norm(points, axis=1)
```

```
# Manual calculation
# distances = np.sqrt(points[:, 0]**2 + points[:, 1]**2)

closest_idx = np.argmin(distances)
closest_point = points[closest_idx]
```