NUMPY TRICKS

📌 np.sort

Use: Sorts an array in ascending order.
Syntax: np.sort(array)
Example:

```
a = np.array([3, 1, 2])

np.sort(a)  # Output: [1 2 3]
```

Why it matters: Useful for ranking, ordering data, or preparing for binary search.

📌 np.append

Use: Adds values to the end of an array.
Syntax: np.append(arr, values)
Example:

```
a = np.array([1, 2, 3])

np.append(a, 4)  # Output: [1 2 3 4]
```

Why it matters: Helps to grow arrays dynamically, like adding new data points.

📌 np.concatenate

Use: Joins two or more arrays along an existing axis.
Syntax: np.concatenate((arr1, arr2), axis=0)
Example:

```
a = np.array([1, 2])

b = np.array([3, 4])

np.concatenate((a, b))  # Output: [1 2 3 4]
```

Why it matters: Essential for merging datasets.

📌 np.unique

Use: Finds unique elements of an array.
Syntax: np.unique(arr)
Example:

```
a = np.array([1, 2, 2, 3])
```

```
np.unique(a)  # Output: [1 2 3]
```

Why it matters: Useful in deduplication, frequency analysis.

📌 np.expand_dims

Use: Adds a new axis (dimension) to an array.
Syntax: np.expand_dims(arr, axis)
Example:

```
a = np.array([1, 2, 3])
```

```
np.expand_dims(a, axis=0)  # Output: [[1 2 3]]
```

Why it matters: Useful for broadcasting or feeding into ML models.

📌 np.where

Use: Returns indices where condition is true.
Syntax: np.where(condition)
Example:

```
a = np.array([1, 2, 3])
```

```
np.where(a > 1)  # Output: (array([1, 2]),)
```

Why it matters: Very helpful in conditional filtering.

📌 np.argmax, np.argmin

Use: Returns index of max or min value.
Syntax: np.argmax(arr), np.argmin(arr)
Example:

```
a = np.array([1, 3, 2])
```

```
np.argmax(a)  # Output: 1

np.argmin(a)  # Output: 0
```

Why it matters: Useful in optimization problems.

📌 np.cumsum, np.cumprod

Use: Cumulative sum or product.
Syntax: np.cumsum(arr), np.cumprod(arr)
Example:

```
a = np.array([1, 2, 3])

np.cumsum(a)   # Output: [1 3 6]

np.cumprod(a)  # Output: [1 2 6]
```

Why it matters: Tracks cumulative growth or progression.

📌 np.percentile

Use: Computes the nth percentile of data.
Syntax: np.percentile(arr, q)
Example:

```
a = np.array([1, 2, 3, 4])

np.percentile(a, 50)  # Output: 2.5
```

Why it matters: Used in statistics and outlier detection.

📌 np.histogram

Use: Computes histogram (frequency distribution).
Syntax: np.histogram(arr, bins)
Example:

```
a = np.array([1, 1, 2, 2, 3])

np.histogram(a, bins=3)

# Output: (array([2, 2, 1]), array([1., 1.666..., 2.333..., 3.]))
```

Why it matters: Helps analyze distribution of data.

📌 np.corrcoef

Use: Computes Pearson correlation coefficient matrix.
Syntax: np.corrcoef(arr1, arr2)
Example:

```
x = np.array([1, 2, 3])

y = np.array([1, 2, 3])

np.corrcoef(x, y)

# Output: 2x2 correlation matrix
```

Why it matters: Measures linear relationship between variables.

📌 np.isin

Use: Checks if elements of one array exist in another.
Syntax: np.isin(arr1, arr2)
Example:

```
np.isin([1, 2, 3], [2, 3, 4])  # Output: [False  True  True]
```

Why it matters: Helpful for filtering based on membership.

📌 np.flip

Use: Reverses the order of elements in an array.
Syntax: np.flip(arr, axis)
Example:

```
a = np.array([1, 2, 3])

np.flip(a)  # Output: [3 2 1]
```

Why it matters: Data reversal, image flipping.

📌 np.put

Use: Replaces elements at specific indices.
Syntax: np.put(arr, indices, values)
Example:

```
a = np.array([0, 1, 2])

np.put(a, [0, 2], [5, 10])  # a becomes [5 1 10]
```

Why it matters: Direct manipulation of specific elements.

📌 np.delete

Use: Deletes elements from array.
Syntax: np.delete(arr, indices)
Example:

```
a = np.array([1, 2, 3])

np.delete(a, 1)  # Output: [1 3]
```

Why it matters: Useful for removing outliers or cleaning data.

✅ Set Functions in NumPy

📌 np.union1d

Use: Union of two arrays (unique values from both).
Example:

```
np.union1d([1, 2], [2, 3])  # Output: [1 2 3]
```

📌 np.intersect1d

Use: Intersection (common values).
Example:

```
np.intersect1d([1, 2], [2, 3])  # Output: [2]
```

📌 np.setdiff1d

Use: Values in arr1 not in arr2.
Example:

```
np.setdiff1d([1, 2], [2, 3])  # Output: [1]
```

📌 np.setxor1d

Use: Elements in only one of the arrays (XOR).
Example:

```
np.setxor1d([1, 2], [2, 3])  # Output: [1 3]
```

📌 np.in1d

Use: Returns boolean array: Is element in another array?
Example:

```
np.in1d([1, 2, 3], [2, 3])  # Output: [False  True  True]
```

📌 np.clip

Use: Limits values to a specified range.
Syntax: np.clip(arr, min, max)
Example:

```
a = np.array([1, 5, 10])

np.clip(a, 2, 6)  # Output: [2 5 6]
```

Why it matters: Prevents values from going out of bounds.
