<https://www.w3resource.com/python-exercises/numpy/index-array.php>

## NumPy Tutorial

[NumPy HOME](https://www.w3schools.com/python/numpy/default.asp" \t "https://www.w3schools.com/python/numpy/_top)[NumPy Intro](https://www.w3schools.com/python/numpy/numpy_intro.asp" \t "https://www.w3schools.com/python/numpy/_top)[NumPy Getting Started](https://www.w3schools.com/python/numpy/numpy_getting_started.asp" \t "https://www.w3schools.com/python/numpy/_top)[NumPy Creating Arrays](https://www.w3schools.com/python/numpy/numpy_creating_arrays.asp" \t "https://www.w3schools.com/python/numpy/_top)[NumPy Array Indexing](https://www.w3schools.com/python/numpy/numpy_array_indexing.asp" \t "https://www.w3schools.com/python/numpy/_top)[NumPy Array Slicing](https://www.w3schools.com/python/numpy/numpy_array_slicing.asp" \t "https://www.w3schools.com/python/numpy/_top)[NumPy Data Types](https://www.w3schools.com/python/numpy/numpy_data_types.asp" \t "https://www.w3schools.com/python/numpy/_top)[NumPy Copy vs View](https://www.w3schools.com/python/numpy/numpy_copy_vs_view.asp" \t "https://www.w3schools.com/python/numpy/_top)[NumPy Array Shape](https://www.w3schools.com/python/numpy/numpy_array_shape.asp" \t "https://www.w3schools.com/python/numpy/_top)[NumPy Array Reshape](https://www.w3schools.com/python/numpy/numpy_array_reshape.asp" \t "https://www.w3schools.com/python/numpy/_top)[NumPy Array Iterating](https://www.w3schools.com/python/numpy/numpy_array_iterating.asp" \t "https://www.w3schools.com/python/numpy/_top)[NumPy Array Join](https://www.w3schools.com/python/numpy/numpy_array_join.asp" \t "https://www.w3schools.com/python/numpy/_top)[NumPy Array Split](https://www.w3schools.com/python/numpy/numpy_array_split.asp" \t "https://www.w3schools.com/python/numpy/_top)[NumPy Array Search](https://www.w3schools.com/python/numpy/numpy_array_search.asp" \t "https://www.w3schools.com/python/numpy/_top)[NumPy Array Sort](https://www.w3schools.com/python/numpy/numpy_array_sort.asp" \t "https://www.w3schools.com/python/numpy/_top)[NumPy Array Filter](https://www.w3schools.com/python/numpy/numpy_array_filter.asp" \t "https://www.w3schools.com/python/numpy/_top)

## NumPy Random

[Random Intro](https://www.w3schools.com/python/numpy/numpy_random.asp" \t "https://www.w3schools.com/python/numpy/_top)[Data Distribution](https://www.w3schools.com/python/numpy/numpy_random_distribution.asp" \t "https://www.w3schools.com/python/numpy/_top)[Random Permutation](https://www.w3schools.com/python/numpy/numpy_random_permutation.asp" \t "https://www.w3schools.com/python/numpy/_top)[Seaborn Module](https://www.w3schools.com/python/numpy/numpy_random_seaborn.asp" \t "https://www.w3schools.com/python/numpy/_top)[Normal Distribution](https://www.w3schools.com/python/numpy/numpy_random_normal.asp" \t "https://www.w3schools.com/python/numpy/_top)[Binomial Distribution](https://www.w3schools.com/python/numpy/numpy_random_binomial.asp" \t "https://www.w3schools.com/python/numpy/_top)[Poisson Distribution](https://www.w3schools.com/python/numpy/numpy_random_poisson.asp" \t "https://www.w3schools.com/python/numpy/_top)[Uniform Distribution](https://www.w3schools.com/python/numpy/numpy_random_uniform.asp" \t "https://www.w3schools.com/python/numpy/_top)[Logistic Distribution](https://www.w3schools.com/python/numpy/numpy_random_logistic.asp" \t "https://www.w3schools.com/python/numpy/_top)[Multinomial Distribution](https://www.w3schools.com/python/numpy/numpy_random_multinomial.asp" \t "https://www.w3schools.com/python/numpy/_top)[Exponential Distribution](https://www.w3schools.com/python/numpy/numpy_random_exponential.asp" \t "https://www.w3schools.com/python/numpy/_top)[Chi Square Distribution](https://www.w3schools.com/python/numpy/numpy_random_chisquare.asp" \t "https://www.w3schools.com/python/numpy/_top)[Rayleigh Distribution](https://www.w3schools.com/python/numpy/numpy_random_rayleigh.asp" \t "https://www.w3schools.com/python/numpy/_top)[Pareto Distribution](https://www.w3schools.com/python/numpy/numpy_random_pareto.asp" \t "https://www.w3schools.com/python/numpy/_top)[Zipf Distribution](https://www.w3schools.com/python/numpy/numpy_random_zipf.asp" \t "https://www.w3schools.com/python/numpy/_top)

Numpy

Key feturas

Mathematical functions

Broadcasting

Broadcasting: Broadcasting is a powerful mechanism that allows NumPy to work with arrays of different shapes when performing arithmetic operations.

Linear Algebra: NumPy provides functions for performing linear algebra operations, including matrix multiplication, eigenvalue computations, and solving linear equations.

Random Number Generation: It includes tools for generating random. numbers from various distributions, which is useful for simulations and probabilistic computations.

Installationi To install NumPy, you. can use pip, Python's the following package installer, by running the command in your terminal or command prompt: pip install numpy

To prepare for interviews using **NumPy**, let’s use a **real-world dataset** and implement **all essential NumPy concepts** with practical examples.

### ✅ Project: ****Air Quality Analysis using NumPy****

#### 📁 Dataset

We will use a real-world **Air Quality Data (CSV)** file that includes:

Date, Time

CO(GT) – Carbon monoxide concentration

NOx(GT) – Nitrogen Oxides concentration

NO2(GT) – Nitrogen Dioxide concentration

📌 [Dataset Download (UCI)](https://archive.ics.uci.edu/ml/machine-learning-databases/00360/AirQualityUCI.zip)

### ✅ NumPy Concepts to Cover

| **Concept** | **Interview Relevance** |
| --- | --- |
| Array creation | Data handling |
| Data types | Memory optimization |
| Indexing & Slicing | Row/column access |
| Broadcasting | Vectorized operations |
| Boolean indexing | Filtering |
| Aggregation | Analysis |
| Reshaping | Preprocessing |
| Stacking | Combining datasets |
| NaN handling | Cleaning |
| Mathematical functions | Transformation |
| Sorting & Searching | Reports |

### ✅ Step-by-Step NumPy Implementation

import numpy as np

import pandas as pd

# Load the dataset

df = pd.read\_csv('AirQualityUCI.csv', sep=';', decimal=',', usecols=range(15))

# Clean the dataset (drop NaNs, remove headers)

df.dropna(how='all', inplace=True)

df = df.drop(columns=['Unnamed: 15', 'Unnamed: 16'], errors='ignore')

df.replace(-200, np.nan, inplace=True)

# Convert to NumPy array

data = df[['CO(GT)', 'NOx(GT)', 'NO2(GT)']].values

### 🔍 Now Implement All Core NumPy Concepts

#### 1. ****Array Creation & Data Types****

print(data.dtype) # float64

print(data.shape) # (9357, 3)

#### 2. ****Basic Stats****

print(np.mean(data, axis=0)) # Mean for each pollutant

print(np.std(data, axis=0)) # Standard deviation

print(np.min(data, axis=0)) # Min values

print(np.max(data, axis=0)) # Max values

#### 3. ****Handling Missing Values****

# Count missing

print(np.isnan(data).sum(axis=0))

# Replace NaN with column mean

means = np.nanmean(data, axis=0)

data = np.where(np.isnan(data), means, data)

#### 4. ****Indexing and Slicing****

# First 5 rows

print(data[:5])

# CO values only

co\_values = data[:, 0]

# Every alternate row

print(data[::2])

#### 5. ****Boolean Indexing****

# Find rows where CO > 2

high\_co = data[data[:, 0] > 2]

print(high\_co.shape)

#### 6. ****Broadcasting****

# Normalize (Min-Max)

min\_vals = np.min(data, axis=0)

max\_vals = np.max(data, axis=0)

normalized = (data - min\_vals) / (max\_vals - min\_vals)

#### 7. ****Reshaping & Stacking****

# Reshape 1D array to column

co\_reshaped = co\_values.reshape(-1, 1)

# Horizontal stack: Add a new computed column (e.g., total pollution)

total\_pollution = np.sum(data, axis=1).reshape(-1, 1)

data\_augmented = np.hstack([data, total\_pollution])

#### 8. ****Mathematical Operations****

# Apply exponential decay model (for example)

decay = np.exp(-data / 10)

#### 9. ****Sorting****

# Sort data by NO2 levels (column 2)

sorted\_data = data[data[:, 2].argsort()]

#### 10. ****Unique, Search, Count****

# Unique CO levels (rounded)

co\_unique = np.unique(np.round(data[:, 0], 1))

# Count how many times a rounded CO value appears

co\_counts = np.bincount(np.round(data[:, 0]).astype(int))

### 📌 Summary

You have now implemented nearly **all essential NumPy operations** that can be asked in interviews using a **real-world air quality dataset**.

### 🧠 Bonus: Practice Questions for Interview

What is broadcasting in NumPy? Show example using data.

How to handle NaN in NumPy arrays?

How to normalize a dataset with NumPy?

What is the difference between np.vstack() and np.hstack()?

How can you select every nth row in a NumPy array?

### 💡 Want More?

I can give:

Interview MCQs on NumPy

Pandas + NumPy combined mini-project

Data visualization for this dataset

Perfect! Let's now **focus on the most important NumPy functions** commonly asked in interviews and demonstrate each using a **real-world dataset** (Air Quality), as discussed.

### ✅ Real-World Dataset: Air Quality UCI

Columns used: CO(GT), NOx(GT), NO2(GT)  
→ Goal: Show how to use np.where**,** np.vstack**,** np.hstack**,** np.nan\_to\_num**,** np.isnan**,** np.replace **equivalent**, and more.

### 🔁 Step 1: Load and Clean Dataset

import numpy as np

import pandas as pd

# Load dataset

df = pd.read\_csv("AirQualityUCI.csv", sep=';', decimal=',', usecols=range(15))

# Drop empty columns and rows

df.dropna(how="all", inplace=True)

df = df.drop(columns=['Unnamed: 15', 'Unnamed: 16'], errors='ignore')

# Replace -200 with NaN (missing data marker in this dataset)

df.replace(-200, np.nan, inplace=True)

# Select key columns

data = df[['CO(GT)', 'NOx(GT)', 'NO2(GT)']].values

## 🔥 Now Use Top NumPy Functions with Examples

### ✅ 1. np.where()

Conditional replacement or selection

# Replace all CO values > 5 with 5 (cap it)

capped\_data = np.where(data[:, 0] > 5, 5, data[:, 0])

# Find indices where NOx > 100

nox\_high\_index = np.where(data[:, 1] > 100)

### ✅ 2. np.isnan() and np.nan\_to\_num() (fillna equivalent)

Handling missing values (NaN)

# Count missing values

missing\_count = np.isnan(data).sum(axis=0)

# Fill missing with 0 (like fillna(0))

data\_no\_nan = np.nan\_to\_num(data, nan=0.0)

### ✅ 3. Replace NaN with column mean

Like Pandas .fillna(df.mean()) but in NumPy

# Compute column means (ignoring NaN)

means = np.nanmean(data, axis=0)

# Replace NaNs with means

data\_filled = np.where(np.isnan(data), means, data)

### ✅ 4. np.vstack() – Vertical Stack (add rows)

# Add a new observation (row)

new\_row = np.array([[1.0, 25.0, 12.0]])

data\_vstacked = np.vstack([data\_filled, new\_row])

### ✅ 5. np.hstack() – Horizontal Stack (add columns)

# Add a new column: sum of all pollutants

total\_pollution = np.sum(data\_filled, axis=1).reshape(-1, 1)

data\_hstacked = np.hstack([data\_filled, total\_pollution])

### ✅ 6. np.unique() with return\_counts

# Unique CO levels and their counts

unique\_co, co\_counts = np.unique(data\_filled[:, 0], return\_counts=True)

### ✅ 7. Replace values (no built-in replace like Pandas, so use np.where)

# Replace CO values == 0 with mean CO

mean\_co = means[0]

co\_replaced = np.where(data\_filled[:, 0] == 0, mean\_co, data\_filled[:, 0])

### ✅ 8. np.round(), np.clip(), np.argmax(), np.argsort()

rounded\_data = np.round(data\_filled, 2)

# Clip NO2 values to a max of 200

data\_clipped = np.copy(data\_filled)

data\_clipped[:, 2] = np.clip(data\_clipped[:, 2], None, 200)

# Row with highest total pollution

max\_index = np.argmax(total\_pollution)

highest\_pollution\_row = data\_filled[max\_index]

## 🧠 Interview-Ready Summary of Must-Know Functions

| **Function** | **Use Case** |
| --- | --- |
| np.where() | Conditional filtering or replacement |
| np.isnan() | Detect NaN values |
| np.nan\_to\_num() | Replace NaNs with specific values |
| np.vstack() | Add row(s) to array vertically |
| np.hstack() | Add column(s) to array horizontally |
| np.unique() | Count unique values |
| np.clip() | Limit values to min/max bounds |
| np.argmax() | Index of max value |
| np.argsort() | Get sort order indices |
| np.round() | Round values to desired precision |

### 📌 Want Practice Task?

Would you like me to give:

MCQs on these NumPy functions?

Small exercises for hands-on practice?

Integration with Pandas/Matplotlib for end-to-end analysis?

Let me know!