# numpy provides lot of inbuilt functions to deal with matrixs and arrays

**Array Creation Routines**

* np.array: Create an array.
* np.asarray: Convert input to an array.
* np.zeros: Create an array of zeros.
* np.ones: Create an array of ones.
* np.empty: Create an uninitialized array.
* np.full: Create an array filled with a specified value.
* np.arange: Create an array with a range of values.
* np.linspace: Create an array with evenly spaced values.
* np.logspace: Create an array with logarithmically spaced values.
* np.eye: Create an identity matrix.
* np.identity: Create an identity matrix.
* np.random.rand: Create an array of random values.
* np.random.randn: Create an array of random values from the standard normal distribution.
* np.random.randint: Create an array of random integers.
* np.fromfunction: Construct an array by executing a function over each coordinate.
* np.fromfile: Create an array from data in a binary file.
* np.frombuffer: Create an array from a buffer.
* np.fromiter: Create an array from an iterator.
* np.fromstring: Create an array from a string.

**Array Manipulation Routines**

* np.reshape: Reshape an array.
* np.ravel: Flatten an array.
* np.flatten: Return a copy of the array collapsed into one dimension.
* np.transpose: Transpose an array.
* np.swapaxes: Swap two axes of an array.
* np.moveaxis: Move axes of an array to new positions.
* np.squeeze: Remove single-dimensional entries from the shape of an array.
* np.expand\_dims: Expand the shape of an array.
* np.concatenate: Join a sequence of arrays along an existing axis.
* np.stack: Join a sequence of arrays along a new axis.
* np.split: Split an array into multiple sub-arrays.
* np.hsplit: Split an array into multiple sub-arrays horizontally.
* np.vsplit: Split an array into multiple sub-arrays vertically.
* np.dsplit: Split an array into multiple sub-arrays along the third axis.

**Arithmetic Operations**

* np.add: Element-wise addition.
* np.subtract: Element-wise subtraction.
* np.multiply: Element-wise multiplication.
* np.divide: Element-wise division.
* np.power: Element-wise power.
* np.mod: Element-wise modulus.
* np.remainder: Equivalent to np.mod.
* np.sqrt: Element-wise square root.
* np.exp: Element-wise exponential.
* np.log: Element-wise natural logarithm.
* np.log10: Element-wise base-10 logarithm.
* np.log2: Element-wise base-2 logarithm.
* np.sin: Element-wise sine.
* np.cos: Element-wise cosine.
* np.tan: Element-wise tangent.
* np.arcsin: Element-wise inverse sine.
* np.arccos: Element-wise inverse cosine.
* np.arctan: Element-wise inverse tangent.

**Statistical Functions**

* np.mean: Compute the arithmetic mean.
* np.median: Compute the median.
* np.std: Compute the standard deviation.
* np.var: Compute the variance.
* np.min: Find the minimum value.
* np.max: Find the maximum value.
* np.sum: Compute the sum.
* np.prod: Compute the product.
* np.cumsum: Compute the cumulative sum.
* np.cumprod: Compute the cumulative product.
* np.percentile: Compute the percentile.
* np.quantile: Compute the quantile.

**Linear Algebra Functions**

* np.dot: Dot product of two arrays.
* np.vdot: Dot product of two vectors.
* np.inner: Inner product of two arrays.
* np.outer: Outer product of two vectors.
* np.matmul: Matrix product of two arrays.
* np.tensordot: Tensor dot product along specified axes.
* np.linalg.solve: Solve a linear matrix equation.
* np.linalg.inv: Compute the (multiplicative) inverse of a matrix.
* np.linalg.det: Compute the determinant of a matrix.
* np.linalg.eig: Compute the eigenvalues and right eigenvectors of a square array.
* np.linalg.svd: Singular Value Decomposition.

**Array Comparison**

* np.equal: Element-wise equality comparison.
* np.not\_equal: Element-wise inequality comparison.
* np.greater: Element-wise greater-than comparison.
* np.greater\_equal: Element-wise greater-than-or-equal comparison.
* np.less: Element-wise less-than comparison.
* np.less\_equal: Element-wise less-than-or-equal comparison.

**Logical Operations**

* np.logical\_and: Element-wise logical AND.
* np.logical\_or: Element-wise logical OR.
* np.logical\_not: Element-wise logical NOT.
* np.logical\_xor: Element-wise logical XOR.

**Miscellaneous Functions**

* np.where: Return elements chosen from x or y depending on condition.
* np.select: Return an array drawn from elements in choicelist, depending on conditions.
* np.choose: Construct an array from an index array and a list of arrays to choose from.
* np.argmax: Return the indices of the maximum values.
* np.argmin: Return the indices of the minimum values.
* np.argsort: Returns the indices that would sort an array.
* np.nonzero: Return the indices of the elements that are non-zero.
* np.count\_nonzero: Count the number of non-zero elements.

**Examples**

Here are a few examples to illustrate some of the functions:

python

Copy code

import numpy as np

# np.asarray example

a = [1, 2, 3]

arr = np.asarray(a)

print(arr) # Output: [1 2 3]

# np.frombuffer example

buffer = b'\x01\x02\x03\x04'

arr = np.frombuffer(buffer, dtype=np.uint8)

print(arr) # Output: [1 2 3 4]

# np.arange example

arr = np.arange(0, 10, 2)

print(arr) # Output: [0 2 4 6 8]

# np.linspace example

arr = np.linspace(0, 1, 5)

print(arr) # Output: [0. 0.25 0.5 0.75 1. ]

# np.add example

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

b = np.array([4, 5, 6])

c = np.add(a, b)

print(c) # Output: [5 7 9]

# np.mean example

a = np.array([1, 2, 3, 4, 5])

mean = np.mean(a)

print(mean) # Output: 3.0

# np.dot example

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

b = np.array([[5, 6], [7, 8]])

dot\_product = np.dot(a, b)

print(dot\_product) # Output: [[19 22]

# [43 50]]

This list provides a solid overview of many of the functions available in NumPy. The library has a rich set of features for handling numerical data efficiently, making it a cornerstone of scientific computing in Python.