

Useful NumPy functions for numerical computing and data manipulation:

Array Creation

- `np.array()` - Create an array from lists or tuples.
- `np.zeros(shape)` - Create an array filled with zeros.
- `np.ones(shape)` - Create an array filled with ones.
- `np.full(shape, fill_value)` - Create an array filled with a specific value.
- `np.eye(n)` - Create an identity matrix.
- `np.arange(start, stop, step)` - Create evenly spaced values within a range.
- `np.linspace(start, stop, num)` - Create linearly spaced values over an interval.
- `np.random.rand(d0, d1, ...)` - Create arrays with random floats between 0 and 1.
- `np.random.randint(low, high, size)` - Create arrays with random integers.

Array Manipulation

- `np.reshape(array, new_shape)` - Change the shape of an array.
- `np.flatten()` - Flatten a multi-dimensional array to 1D.
- `np.transpose(array)` - Transpose the dimensions of an array.
- `np.concatenate()` - Join two or more arrays.
- `np.vstack()`, `np.hstack()` - Stack arrays vertically or horizontally.
- `np.split(array, indices_or_sections)` - Split array into multiple sub-arrays.

Mathematical Functions (Universal Functions - ufuncs)

- `np.add()`, `np.subtract()`, `np.multiply()`, `np.divide()` - Basic arithmetic.
- `np.power()`, `np.mod()` - Exponentiation and modulus.
- `np.sin()`, `np.cos()`, `np.tan()`, `np.sinh()`, `np.cosh()`, `np.tanh()` - Trigonometric functions.
- `np.exp()`, `np.log()`, `np.log10()`, `np.log2()`, `np.sqrt()` - Exponential and logarithmic functions.
- `np.abs()` - Absolute value.
- `np.round()`, `np.floor()`, `np.ceil()` - Rounding functions.
- `np.clip(array, min, max)` - Limit values in array within min and max.

Aggregation Functions

- `np.sum(x, axis=None)` — Sum of array elements over a given axis
- `np.prod(x, axis=None)` — Product of array elements
- `np.mean(x, axis=None)` — Arithmetic mean along an axis
- `np.median(x, axis=None)` — Median along an axis
- `np.std(x, axis=None)` — Standard deviation
- `np.var(x, axis=None)` — Variance
- `np.min(x, axis=None)` — Minimum value
- `np.max(x, axis=None)` — Maximum value
- `np.argmin(x, axis=None)` — Index of minimum value
- `np.argmax(x, axis=None)` — Index of maximum value
- `np.percentile(x, q, axis=None)` — Compute data percentiles
- `np.any(x, axis=None)` — Test if any array element along axis evaluates to True
- `np.all(x, axis=None)` — Test if all array elements are True

Statistical Functions

- `np.mean()`, `np.median()`, `np.std()`, `np.var()` - Mean, median, standard deviation, variance.
- `np.min()`, `np.max()`, `np.argmin()`, `np.argmax()` - Minimum, maximum and their indices.
- `np.percentile()` - Compute percentile ranks.
- `np.corrcoef()` - Correlation coefficient matrix.

Linear Algebra

- `np.dot()` - Dot product of two arrays.
- `np.matmul()` - Matrix multiplication.

- `np.linalg.inv()` - Inverse of a matrix.
- `np.linalg.det()` - Determinant of a matrix.
- `np.linalg.eig()` - Eigenvalues and eigenvectors.

Logical and Comparison

- `np.all()`, `np.any()` - Check if all or any element is True.
- `np.where()` - Return indices where condition is True.
- `np.logical_and()`, `np.logical_or()`, `np.logical_not()` - Logical operations.
- `np.equal()`, `np.greater()`, `np.less()` - Comparison operations.

Others

- `np.unique()` - Find unique elements.
- `np.sort()` - Sort the array.
- `np.argsort()` - Indices that would sort an array.
- `np.copy()` - Create a copy of an array.

Pandas

Pandas is a powerful and popular Python library designed for data manipulation and analysis, often used in data science and analytics.

Key Features of Pandas:

- **Data Structures:** Provides Series (1D labeled array) and DataFrame (2D labeled table) for handling structured data.
- **DataFrame Creation:** Easily create DataFrames from dictionaries, lists, CSV files, Excel files, and more.
- **Data Access:** Use `.loc[]` (label-based) and `.iloc[]` (integer position-based) to select data by rows and columns.
- **Data Manipulation:** Add, remove, rename, and modify columns and rows effortlessly.
- **Filtering:** Filter data using conditions and complex queries.
- **Missing Data Handling:** Detect, fill, or drop missing values with methods like `.isna()`, `.fillna()`, and `.dropna()`.
- **Aggregation:** Group data using `.groupby()` and aggregate with `.sum()`, `.mean()`, `.count()`, etc.
- **Merging and Joining:** Combine DataFrames like SQL joins using `.merge()` and `.concat()`.
- **Sorting and Ranking:** Sort datasets by one or more columns and rank elements.
- **Apply Functions:** Use `.apply()` to apply custom or predefined functions across data.
- **I/O Capabilities:** Read/write data from/to various formats (CSV, Excel, SQL, JSON, etc.).

Example

python

```
import pandas as pd

# Create a DataFrame
df = pd.DataFrame({
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'City': ['NY', 'LA', 'Chicago']
})

# Access and filter data
print(df.loc[df['Age'] > 28])

# Add new column
df['Country'] = 'USA'

# Calculate mean age
mean_age = df['Age'].mean()
```