KANIKA CHOUHAN

Key Numpy Functions in Data Analysis

kanikachouhan1515

kanikachouhan1515@gmail.com

KANIKA CHOUHAN

1

1. Array Creation

- **np.array():** Converts a list, tuple, or other data structure into a NumPy array. It forms the foundation for working with arrays in NumPy.
- **np.zeros():** Creates an array filled with zeros. Commonly used to initialize arrays when you know the shape but not the content.
- **np.ones():** Similar to zeros(), but fills the array with ones instead. Useful for initializing arrays where a default value of 1 is required.
- **np.arange():** Generates an array of evenly spaced values within a given range. Useful for creating sequences of numbers.
- **np.linspace():** Generates a specified number of evenly spaced numbers over a given interval, useful for defining ranges with a specific number of elements.

kanikachouhan1515
kanikachouhan1515@gmail.com

KANIKA CHOUHAN

```python
import numpy as np

# Create a 1D array from a list
arr = np.array([1, 2, 3, 4, 5])

# Create an array of zeros
arr_zeros = np.zeros((3, 3))

# Create an array of ones
arr_ones = np.ones((2, 4))

# Create an array with a range of numbers
arr_range = np.arange(0, 10, 2)

# Create an array of evenly spaced numbers
arr_linspace = np.linspace(0, 1, 5)
```

kanikachouhan1515

kanikachouhan1515@gmail.com

KANIKA CHOUHAN

2

2.Mathematical Operations

- **np.add():** Performs element-wise addition of two arrays.
- np.subtract(): Performs element-wise subtraction of one array from another.
- **np.multiply():** Multiplies elements of two arrays element by element for scalar or matrix multiplication.
- **np.divide():** Divides elements of one array by the corresponding elements in another.
- **np.exp():** Computes the exponential (e^x) of each element in the array.
- **np.sqrt():** Computes the square root of each element in the array..
- **np.log():** Computes the natural logarithm (base e) of each element in the array.

kanikachouhan1515

kanikachouhan1515@gmail.com

KANIKA CHOUHAN

```
# Element-wise addition
arr_sum = np.add(arr1, arr2)

# Element-wise subtraction
arr_diff = np.subtract(arr1, arr2)

# Element-wise multiplication
arr_mult = np.multiply(arr1, arr2)

# Element-wise division
arr_div = np.divide(arr1, arr2)

# Exponentiation
arr_exp = np.exp(arr)

# Square root of each element
arr_sqrt = np.sqrt(arr)

# Logarithm of each element
arr_log = np.log(arr)
```

kanikachouhan1515

kanikachouhan1515@gmail.com

KANIKA CHOUHAN

3.Statistical Functions

- **np.mean():** Calculates the arithmetic mean (average) of the array elements. It's useful for summarizing the central tendency of data.
- **np.median():** Finds the median of the array, which is helpful when you want to analyze the midpoint of the data distribution.
- **np.std():** Computes the standard deviation, a measure of how spread out the data is. Often used for variability analysis.
- **np.var():** Calculates the variance, another measure of the spread of data, often used in conjunction with standard deviation.
- **np.min():** Finds the minimum value within the array,
- **np.max():** Finds the maximum value in the array,
- **np.sum():** Adds up all the elements in the array.

kanikachouhan1515
kanikachouhan1515@gmail.com

KANIKA CHOUHAN

```python
# Mean of the array
mean_val = np.mean(arr)

# Median of the array
median_val = np.median(arr)

# Standard deviation
std_val = np.std(arr)

# Variance of the array
var_val = np.var(arr)

# Minimum value in the array
min_val = np.min(arr)

# Maximum value in the array
max_val = np.max(arr)

# Sum of all elements in the array
total_sum = np.sum(arr)
```

kanikachouhan1515

kanikachouhan1515@gmail.com

KANIKA CHOUHAN

4

4. Array Reshaping

- **np.reshape():** Reshapes an array I. e. modify the dimensions without changing its data,
- **np.flatten():** Converts a multi-dimensional array into a one-dimensional array.
- **np.concatenate():** Joins two or more arrays along an existing axis.
- **np.vstack():** Stacks arrays vertically (row-wise), useful for combining arrays into a larger 2D array.
- **np.hstack():** Stacks arrays horizontally (column-wise), helping to append arrays side-by-side for analysis.
- **np.transpose():** Swaps the axes of an array
- **np.sort():** Sorts the elements of an array,

kanikachouhan1515
kanikachouhan1515@gmail.com

KANIKA CHOUHAN

```
# Reshape array
arr_reshaped = np.reshape(arr, (3, 2))

# Flatten the array to 1D
arr_flattened = arr.flatten()

# Concatenate two arrays
arr_concat = np.concatenate((arr1, arr2))

# Stack arrays vertically (row-wise)
arr_vstack = np.vstack((arr1, arr2))

# Stack arrays horizontally (column-wise)
arr_hstack = np.hstack((arr1, arr2))

# Transpose of the array
arr_transpose = np.transpose(arr)

# Sort elements in the array
arr_sorted = np.sort(arr)
```

kanikachouhan1515

kanikachouhan1515@gmail.com

KANIKA CHOUHAN

5. Random No. Generation

- **np.random.rand():** Generates an array of random floats between 0 and 1.
- **np.random.randint():** Generates random integers within a specified range.
- **np.random.randn():** Produces random numbers from the standard normal distribution (mean=0, std=1), essential for modeling statistical processes.
- **np.random.shuffle():** Randomly shuffles the elements of an array along the first axis.
- **np.random.choice():** Selects random elements from an array.

kanikachouhan1515

kanikachouhan1515@gmail.com

KANIKA CHOUHAN

```python
# Generate random numbers between 0 and 1
rand_arr = np.random.rand(3, 3)

# Generate random integers within a range
rand_int_arr = np.random.randint(0, 10, size=(3, 3))

# Generate random numbers from a normal distribution
rand_normal_arr = np.random.randn(3, 3)

# Shuffle the array in-place
np.random.shuffle(arr)

# Generate a random choice from an array
rand_choice = np.random.choice(arr)
```

kanikachouhan1515

kanikachouhan1515@gmail.com

KANIKA CHOUHAN

6. Linear Algebra

- **np.dot():** Computes the dot product between two arrays (vectors or matrices).
- **np.matmul():** Performs matrix multiplication between two 2D arrays,
- **np.equal():** Compares two arrays element by element and returns a boolean array indicating where elements are equal.
- **np.cumsum():** Returns the cumulative sum of array elements, useful in financial and time-series analysis where cumulative totals matter.
- **np.cumprod():** Returns the cumulative product of array elements, often used in statistical models and simulations.

kanikachouhan1515
kanikachouhan1515@gmail.com

KANIKA CHOUHAN

```
# Dot product of two arrays
dot_product = np.dot(arr1, arr2)

# Matrix multiplication
mat_mult = np.matmul(arr1, arr2)

# Element-wise comparison (returning boolean array)
arr_equal = np.equal(arr1, arr2)

# Cumulative sum of elements
cumsum_arr = np.cumsum(arr)

# Cumulative product of elements
cumprod_arr = np.cumprod(arr)
```

kanikachouhan1515

kanikachouhan1515@gmail.com

KANIKA CHOUHAN

7. Indexing & Slicing

- **np.array[index]:** Accesses elements in an array by index, a basic but essential operation for working with data in NumPy.
- **array[start:end]:** Slices arrays, allowing you to extract subarrays or specific ranges of data.
- **Boolean Indexing:** Filters an array using a condition to return a subset of elements that meet that condition. This is useful for data filtering and cleaning.
- **Fancy Indexing:** Selects specific elements using lists or arrays of indices, providing more flexibility for subsetting data.

kanikachouhan1515

kanikachouhan1515@gmail.com

KANIKA CHOUHAN

```python
# Index a specific element
element = arr[2]

# Slice the array
arr_slice = arr[1:4]

# Boolean indexing (filtering based on a condition)
arr_bool = arr[arr > 2]

# Fancy indexing
arr_fancy = arr[[0, 2, 4]]
```

kanikachouhan1515

kanikachouhan1515@gmail.com

KANIKA CHOUHAN

8. Broadcasting

- Broadcasting:
- A powerful feature that allows NumPy to perform element-wise operations between arrays of different shapes.
- This reduces the need for manual array resizing and simplifies code.
- For example, adding a scalar to an array adds the scalar to each element of the array automatically.

kanikachouhan1515
kanikachouhan1515@gmail.com

KANIKA CHOUHAN

```
# Broadcasting: add a scalar to each element in the array
arr_broadcast = arr + 10
```

kanikachouhan1515

kanikachouhan1515@gmail.com

KANIKA CHOUHAN

FOUND THIS USEFUL ???

LIKE

&

FOLLOW

for more

kanikachouhan1515

kanikachouhan1515@gmail.com

SHARE