No.	Subcategory	Function Name	Arguments	Function Description	Example
1	Importing NumPy	*np.array()*	list or sequence-like: The input list or sequence.	The `np.array()` function is used to create arrays from lists or other sequences.	`array_1d = np.array([1, 2, 3, 4, 5])`
2	Creating Arrays with Zeros	*np.zeros()*	shape: Tuple of integers representing the array's shape.	The `np.zeros()` function creates an array filled with zeros.	`zero_array = np.zeros((2, 3))  # Creates a 2x3 array filled with zeros`
3	Creating Arrays with Ones	*np.ones()*	shape: Tuple of integers representing the array's shape.	The `np.ones()` function creates an array filled with ones.	`ones_array = np.ones((3, 2))  # Creates a 3x2 array filled with ones`
4	Creating Arrays with a Range of Values	*np.arange()*	start: Starting value of the range, stop: End value, step: Interval between values.	The `np.arange()` function creates an array with values from a specified range.	`range_array = np.arange(0, 10, 2)  # Array of numbers from 0 to 10, step 2`
5	Creating Arrays with Linearly Spaced Values	*np.linspace()*	start: Starting value, stop: End value, num: Number of samples to generate.	The `np.linspace()` function creates an array with specified number of values, linearly spaced.	`linspace_array = np.linspace(0, 10, 5)  # 5 numbers between 0 and 10`
6	Creating Random Arrays	*np.random.rand()*	shape: Tuple indicating the shape of the random array.	The `np.random.rand()` function creates an array with random values between 0 and 1.	`random_array = np.random.rand(2, 3)  # 2x3 array with random values`
7	Creating Random Integers	*np.random.randint()*	low: Lower bound, high: Upper bound, size: Shape of the random integers array.	The `np.random.randint()` function generates random integers within a specified range.	`random_int_array = np.random.randint(0, 10, (2, 3))  # Random integers between 0 and 10, shape 2x3`
8	Changing the Shape of an Array	*np.reshape()*	shape: Tuple representing the new shape.	The `np.reshape()` function changes the shape of an existing array.	`reshaped_array = np.arange(9).reshape(3, 3)  # Reshapes a 1D array into a 3x3 array`
9	Concatenating Arrays	*np.concatenate()*	arrays: Tuple of arrays to join, axis: Axis along which to concatenate.	The `np.concatenate()` function combines two or more arrays along a specified axis.	`concatenated_array = np.concatenate((array_1, array_2))`
10	Stacking Arrays Vertically	*np.vstack()*	arrays: Tuple of arrays to stack vertically.	The `np.vstack()` function stacks arrays vertically (row-wise).	`stacked_array = np.vstack((array_1, array_2))`
11	Stacking Arrays Horizontally	*np.hstack()*	arrays: Tuple of arrays to stack horizontally.	The `np.hstack()` function stacks arrays horizontally (column-wise).	`stacked_array = np.hstack((array_1, array_2))`
12	Summing Array Elements	*np.sum()*	array: Array-like, axis: Axis along which the sum is computed.	The `np.sum()` function computes the sum of array elements along a specified axis.	`sum_array = np.sum(array, axis=0)  # Sum along columns`
13	Calculating the Mean	*np.mean()*	array: Array-like.	The `np.mean()` function calculates the mean of array elements.	`mean_value = np.mean(array)`
14	Calculating the Median	*np.median()*	array: Array-like.	The `np.median()` function computes the median of array elements.	`median_value = np.median(array)`
15	Calculating the Standard Deviation	*np.std()*	array: Array-like.	The `np.std()` function calculates the standard deviation.	`std_dev = np.std(array)`
16	Finding the Minimum and Maximum	*np.min(), np.max()*	array: Array-like.	The `np.min()` and `np.max()` functions return the minimum and maximum values of the array, respectively.	`min_value = np.min(array), max_value = np.max(array)`
17	Indices of Minimum and Maximum	*np.argmin(), np.argmax()*	array: Array-like.	The `np.argmin()` and `np.argmax()` functions return the indices of the minimum and maximum values, respectively.	`min_index = np.argmin(array), max_index = np.argmax(array)`

18	Sorting Arrays	*np.sort()*	array: Array-like, axis: Axis along which to sort.	The `np.sort()` function sorts an array in ascending order.	`sorted_array = np.sort(array, axis=None)`
19	Finding Unique Elements	*np.unique()*	array: Array-like.	The `np.unique()` function finds unique elements of an array.	`unique_array = np.unique(array)`
20	Dot Product of Arrays	*np.dot()*	array_1, array_2: Arrays to compute dot product.	The `np.dot()` function computes the dot product of two arrays.	`dot_product = np.dot(array_1, array_2)`
21	Cross Product of Arrays	*np.cross()*	array_1, array_2: Arrays to compute cross product.	The `np.cross()` function computes the cross product of two arrays.	`cross_product = np.cross(array_1, array_2)`
22	Matrix Inversion	*np.linalg.inv()*	matrix: Square matrix to compute the inverse.	The `np.linalg.inv()` function computes the inverse of a matrix.	`inverse_matrix = np.linalg.inv(matrix)`
23	Eigenvalues and Eigenvectors	*np.linalg.eig()*	matrix: Square matrix to compute eigenvalues and eigenvectors.	The `np.linalg.eig()` function computes the eigenvalues and eigenvectors of a matrix.	`eigenvalues, eigenvectors = np.linalg.eig(matrix)`