Common NumPy Array Functions

There are many NumPy array functions available but here are some of the most commonly used ones.

Array Operations	Functions
Array Creation Functions	np.array(), np.zeros(), np.ones(), np.empty(), etc.
Array Manipulation Functions	np.reshape(), np.transpose(), etc.
Array Mathematical Functions	np.add(), np.subtract(), np.sqrt(), np.power(), etc.
Array Statistical Functions	np.median(), np.mean(), np.std(), and np.var().
Array Input and Output Functions	np.save(), np.load(), np.loadtxt(), etc.

NumPy Array Creation Functions

Array creation functions allow us to create new NumPy arrays. For example,

```
import numpy as np

# create an array using np.array()
array1 = np.array([1, 3, 5])
print("np.array():\n", array1)

# create an array filled with zeros using np.zeros()
array2 = np.zeros((3, 3))
print("\nnp.zeros():\n", array2)

# create an array filled with ones using np.ones()
array3 = np.ones((2, 4))
print("\nnp.ones():\n", array3)
```

Output

```
np.array():
[1 3 5]

np.zeros():
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
np.ones():
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]]
```

Here,

- np.array() - creates an array from a Python List
- np.zeros() - creates an array filled with zeros of the specified shape
- np.ones() - creates an array filled with ones of the specified shape

NumPy Array Manipulation Functions

NumPy array manipulation functions allow us to modify or rearrange NumPy arrays. For example,

```
import numpy as np

# create a 1D array
array1 = np.array([1, 3, 5, 7, 9, 11])

# reshape the 1D array into a 2D array
array2 = np.reshape(array1, (2, 3))

# transpose the 2D array
array3 = np.transpose(array2)

print("Original array:\n", array1)
print("\nReshaped array:\n", array2)
print("\nTransposed array:\n", array3)
```

Output

```
Original array:
[ 1  3  5  7  9 11]

Reshaped array:
 [[ 1  3  5]
 [ 7  9 11]]

Transposed array:
 [[ 1  7]
 [ 3  9]
 [ 5 11]]
```

In this example,

- np.reshape(array1, (2, 3)) - reshapes array1 into 2D array with shape (2,3)

- `np.transpose(array2)` - transposes 2D array `array2`

NumPy Array Mathematical Functions
In NumPy, there are tons of mathematical functions to perform on arrays. For example,

```
import numpy as np

# create two arrays
array1 = np.array([1, 2, 3, 4, 5])
array2 = np.array([4, 9, 16, 25, 36])

# add the two arrays element-wise
arr_sum = np.add(array1, array2)

# subtract the array2 from array1 element-wise
arr_diff = np.subtract(array1, array2)

# compute square root of array2 element-wise
arr_sqrt = np.sqrt(array2)


print("\nSum of arrays:\n", arr_sum)
print("\nDifference of arrays:\n", arr_diff)
print("\nSquare root of first array:\n", arr_sqrt)
```

Output

```
Sum of arrays:
[ 5 11 19 29 41]

Difference of arrays:
 [ -3  -7 -13 -21 -31]

Square root of first array:
 [2. 3. 4. 5. 6.]
```

NumPy Array Statistical Functions
NumPy provides us with various statistical functions to perform statistical data analysis. These statistical functions are useful to find basic statistical concepts like mean, median, variance, etc. It is also used to find the maximum or the minimum element in an array.
Let's see an example.

```
import numpy as np

# create a numpy array
marks = np.array([76, 78, 81, 66, 85])

# compute the mean of marks
mean_marks = np.mean(marks)
print("Mean:",mean_marks)

# compute the median of marks
median_marks = np.median(marks)
print("Median:",median_marks)

# find the minimum and maximum marks
min_marks = np.min(marks)
print("Minimum marks:", min_marks)

max_marks = np.max(marks)
print("Maximum marks:", max_marks)
```

Output

```
Mean: 77.2
Median: 78.0
Minimum marks: 66
Maximum marks: 85
```

Here, computed the mean, median, minimum, and maximum of the given array `marks`.

NumPy Array Input/Output Functions
NumPy offers several input/output (I/O) functions for loading and saving data to and from files. For example,

```
import numpy as np

# create an array
array1 = np.array([[1, 3, 5], [2, 4, 6]])

# save the array to a text file
np.savetxt('data.txt', array1)

# load the data from the text file
loaded_data = np.loadtxt('array1.txt')

# print the loaded data
print(loaded_data)
```

Output

```
[[1. 3. 5.]
 [2. 4. 6.]]
```

In this example, we first created the 2D array named `array1` and then saved it to a text file using the `np.savetxt()` function.
We then loaded the saved data using the `np.loadtxt()` function.