Python: NumPy

The library NumPy is used to create and manipulate arrays in python. It is useful as it is providing fast access to data over python lists. The library is imported at the start of the script.

```python
import numpy as np

print(np.__version__)

arr = np.array([1, 2, 3, 4, 5])
print(arr)
print(type(arr))
```

```
2.2.1
[1 2 3 4 5]
<class 'numpy.ndarray'>
```

```python
# 0d
arr = np.array(42)
print(arr)

# 1d
arr = np.array([1, 2, 3, 4, 5])
print(arr)

# 2d
arr = np.array([[1, 2, 3], [4, 5, 6]])
print(arr)

# 3d
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])

print(arr)
# Check Number of Dimensions
a = np.array(42)
b = np.array([1, 2, 3, 4, 5])
c = np.array([[1, 2, 3], [4, 5, 6]])

print(a.ndim)
print(b.ndim)
print(c.ndim)

# Higher Dimensional Arrays
arr = np.array([1, 2, 3, 4], ndmin=5)
print(arr)
print('number of dimensions :', arr.ndim)
```

```
42
[1 2 3 4 5]
[[1 2 3]
[4 5 6]]
[[[1 2 3]
[4 5 6]]

[[1 2 3]
[4 5 6]]]
0
1
2
[[[[[1 2 3 4]]]]]
number of dimensions : 5
```

```python
# Create a 1D array
arr = np.array([1, 2, 3, 4])

# Access the first element
print(arr[0])

# Access the second element
print(arr[1])

# Add the third and fourth elements
print(arr[2] + arr[3])
```

```
1
2
7
```

```python
# Create a 2D array
arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])

# Access the element on the first row, second column
print(arr[0, 1])

# Access the element on the second row, fifth column
print(arr[1, 4])
```

```
2
10
```

```python
# Create a 3D array
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])

# Access the first element of the first 2D array
print(arr[0, 0, 0])

# Access the second element of the second 2D array
print(arr[1, 1, 1])
```

```
1
11
```

```python
# Negative Index
arr = np.array([1, 2, 3, 4])
# Access the last element
print(arr[-1])

# Access the second last element
print(arr[-2])
```

```
4
3
```

```
# Slicing 1-D Arrays
arr = np.array([1, 2, 3, 4, 5, 6, 7])

# Slice elements from index 1 to 5 (2nd to 5th elements)
print(arr[1:5])

# Slice elements from the beginning to index 4 (not included)
print(arr[:4])

# Slice elements from index 4 to the end
print(arr[4:])

#Negative Slicing
arr = np.array([1, 2, 3, 4, 5, 6, 7])

# Slice from the third last to the second last element
print(arr[-3:-1])

# Slicing with Step
arr = np.array([1, 2, 3, 4, 5, 6, 7])

# Slice every second element from index 1 to 5
print(arr[1:5:2])

# Slice every second element from the entire array
print(arr[::2])
```

```
[2 3 4 5]
[1 2 3 4]
[5 6 7]
[5 6]
[2 4]
[1 3 5 7]
```

```
#Slicing 2-D Arrays:
arr = np.array([[1, 2, 3, 4],
                [5, 6, 7, 8],
                [9, 10, 11, 12]])

# Slice elements from the second row, index 1 to 3
print(arr[1, 1:3])

# Slice elements from all rows, index 2
print(arr[:, 2])

# Slice elements from row 1 onwards, column 1 onwards
print(arr[1:, 1:])
```

```
[6 7]
[ 3  7 11]
[[ 6  7  8]
 [10 11 12]]
```

```python
# Integer array
int_array = np.array([1, 2, 3], dtype='i')
print(int_array)
print(int_array.dtype)

# String array
string_array = np.array(['a', 'b', 'c'], dtype='S')
print(string_array)
print(string_array.dtype)
```

```
[1 2 3]
int32
[b'a' b'b' b'c']
|S1
```

```python
# Array Type Conversion
# Integer array
int_array = np.array([1, 2, 3])
print(int_array.dtype)

# Convert to float
float_array = int_array.astype('f')
print(float_array)
print(float_array.dtype)
```

```
int64
[1. 2. 3.]
float32
```

```python
# Creating a Copy
original_array = np.array([1, 2, 3, 4, 5])
copied_array = original_array.copy()
original_array[0] = 99

print("Original Array:", original_array)
print("Copied Array:", copied_array)

# Creating a View
original_array = np.array([1, 2, 3, 4, 5])
view_array = original_array.view()
original_array[0] = 99

print("Original Array:", original_array)
print("View Array:", view_array)

# Modifying the View
original_array = np.array([1, 2, 3, 4, 5])
view_array = original_array.view()
view_array[1] = 88

print("Original Array:", original_array)
print("View Array:", view_array)
```

```
Original Array: [99  2  3  4  5]
Copied Array: [1 2 3 4 5]
Original Array: [99  2  3  4  5]
View Array: [99  2  3  4  5]
Original Array: [ 1 88  3  4  5]
View Array: [ 1 88  3  4  5]
```

```python
# Getting the Shape of an Array
# 2-D array
arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
print(arr.shape)  # Output: (2, 4)

# Creating Arrays with a Specific Shape
# Create a 5-D array
arr = np.array([1, 2, 3, 4], ndmin=5)
print(arr)
print('Shape of array:', arr.shape)

# Reshape from 1-D to 2-D
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
newarr = arr.reshape(4, 3)
print(newarr)

# Reshape from 1-D to 3-D
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
newarr = arr.reshape(2, 3, 2)
print(newarr)

# Flattening Arrays
arr = np.array([[1, 2, 3], [4, 5, 6]])
flat_arr = arr.reshape(-1)
print(flat_arr)
```

```
(2, 4)
[[[[[1 2 3 4]]]]]
Shape of array: (1, 1, 1, 1, 4)
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
[[[ 1  2]
  [ 3  4]
  [ 5  6]]

 [[ 7  8]
  [ 9 10]
  [11 12]]]
[1 2 3 4 5 6]
```

```python
# Iterating 1-D Arrays
arr = np.array([1, 2, 3])
for x in arr:
   print(x)

# Iterating 2-D Arrays
arr = np.array([[1, 2, 3], [4, 5, 6]])
for row in arr:
   print(row)

# Iterating 3-D Arrays
arr = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
for matrix in arr:
   print(matrix)

# Using nditer() for Multi-Dimensional Arrays:
arr = np.array([[1, 2, 3], [4, 5, 6]])
for x in np.nditer(arr):
   print(x)
```

```
1
2
3
[1 2 3]
[4 5 6]
[[1 2]
[3 4]]
[[5 6]
[7 8]]
1
2
3
4
5
6
```

```python
# Using concatenate() Function:
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])

# Join two 1-D arrays
arr = np.concatenate((arr1, arr2))
print(arr)

# Join two 2-D arrays along rows (axis=1)
arr1 = np.array([[1, 2], [3, 4]])
arr2 = np.array([[5, 6], [7, 8]])
arr = np.concatenate((arr1, arr2), axis=1)
print(arr)
```

```
[1 2 3 4 5 6]
[[1 2 5 6]
[3 4 7 8]]
```

```python
# Using stack() Function:
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])

# Stack two 1-D arrays along a new axis (axis=1)
arr = np.stack((arr1, arr2), axis=1)
print(arr)
```

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

```python
# Using split() Function:
arr = np.array([1, 2, 3, 4, 5, 6])

# Split the array into 3 equal sub-arrays
sub_arrays = np.split(arr, 3)
print(sub_arrays)

# Using array_split() Function:
arr = np.array([1, 2, 3, 4, 5, 6])

# Split the array into 4 sub-arrays
sub_arrays = np.array_split(arr, 4)
print(sub_arrays)
```

```
[array([1, 2]), array([3, 4]), array([5, 6])]
[array([1, 2]), array([3, 4]), array([5]), array([6])]
```

```python
# Using where() Function:
arr = np.array([1, 2, 3, 4, 5, 4, 4])

# Find indices where the value is 4
indices = np.where(arr == 4)
print(indices)
```

```
(array([3, 5, 6]),)
```

```
# Using sort() Function:
arr = np.array([3, 1, 2, 5, 4])

# Sort the array
sorted_arr = np.sort(arr)
print(sorted_arr)

# In-Place Sorting with sort() Method:
arr = np.array([3, 1, 2, 5, 4])

# Sort the array in place
arr.sort()
print(arr)
```

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

```
# Filtering with Boolean Indexing:
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

# Create a boolean mask where elements are greater than 5
mask = arr > 5

# Apply the mask to filter elements
filtered_arr = arr[mask]
print(filtered_arr)


# Filtering with Multiple Conditions:
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

# Filter elements greater than 3 and less than 8
filtered_arr = arr[(arr > 3) & (arr < 8)]
print(filtered_arr)
```

```
[ 6  7  8  9 10]
[4 5 6 7]
```

```python
# Custom Probability Distribution:
# Define the population and corresponding probabilities
population = [1, 2, 3, 4, 5]
probabilities = [0.1, 0.2, 0.3, 0.2, 0.2]

# Generate 5 random numbers based on the custom distribution
custom_dist_arr = np.random.choice(population, 5, p=probabilities)
print(custom_dist_arr)


# Normal distribution
# Generate 5 random numbers with mean 0 and standard deviation 1
random_samples = np.random.normal(0, 1, 5)
print(random_samples)


# Binomial Distribution:
# Generate 5 random numbers with 10 trials and probability of success 0.5
binomial_arr = np.random.binomial(10, 0.5, 5)
print(binomial_arr)

# Seaborn Module
import seaborn as sns
import matplotlib.pyplot as plt

data = np.random.normal(loc=0, scale=1, size=1000)

sns.histplot(data, kde=False)
plt.title('Histogram of Random Data')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```

```
[4 4 2 3 5]
[-0.06280635  1.65000601 -0.39743189 -0.83078919 -
1.15834824]
[7 5 7 4 3]
```