numpy

August 9, 2024

[57]:
```python
# Check The Version
import numpy as np

print(np.__version__)
```

```
2.0.0
```

[58]:
```python
# List To Array
testList = [1, 2, 3, 4]
print(type(testList), testList)

listArray = np.array(testList)
print(type(listArray), listArray)
```

```
<class 'list'> [1, 2, 3, 4]
<class 'numpy.ndarray'> [1 2 3 4]
```

[59]:
```python
# Tuple To Array
testTuple = (1, 2, 3, 4, 5)
print(type(testTuple), testTuple)

tupleArray = np.array(testTuple)
print(type(tupleArray), tupleArray)
```

```
<class 'tuple'> (1, 2, 3, 4, 5)
<class 'numpy.ndarray'> [1 2 3 4 5]
```

[60]:
```python
# Set To Array
testSet = {1, 2, 3, 4, 5, 3, 4}
print(type(testSet), testSet)

setArray = np.array(testSet)
print(type(setArray), setArray)
```

```
<class 'set'> {1, 2, 3, 4, 5}
<class 'numpy.ndarray'> {1, 2, 3, 4, 5}
```

[61]:
```
# 0-D Array
array = np.array(28)
print(array.ndim, array)
```

```
0 28
```

[62]:
```
# 1-D Array
array = np.array([1, 2, 3, 4, 5])
print(array.ndim, array)
```

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

[63]:
```
# 2-D Array
array = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])
print(array.ndim, array)
```

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

[64]:
```
# 3-D Array
# An array that has 2-D arrays (matrices) as its elements is called 3-D array.
array = np.array(
    [[[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]], [[0, 0, 0, 0, 0], [1, 1, 1, 1, 1]]]
)
print(array.ndim, array)
```

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

 [[ 0  0  0  0  0]
  [ 1  1  1  1  1]]]
```

[65]:
```
# Access Array
array = np.array([1, 2, 3, 4])
print(array[3])

array = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
print(array[1, 3])

array = np.array([[[1, 2, 3, 4], [4, 3, 2, 1]], [[1, 0, 1, 0], [1, 1, 1, 1]]])
print(array[0, 0, -3])
```

```
4
8
2
```

[66]:
```
# Array Slicing
array = np.array([1, 2, 3, 4, 5, 6, 7, 8])
```

```python
print(array[:-2])
```

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

[67]:
```python
# Data Types
arrayInt = np.array([1, 2, 3, 4])
print(arrayInt.dtype)

arrayString = np.array(["Emn", "Sad", "Ank"])
print(arrayString.dtype)
```

```
int64
<U3
```

[68]:
```python
# Copy Array
array = np.array([1, 2, 3, 4, 5])
X = array.copy()
array[0] = 10

print(array)
print(X)
```

```
[10  2  3  4  5]
[1 2 3 4 5]
```

[69]:
```python
# View Array
array = np.array([1, 2, 3, 4, 5])
X = array.view()
array[0] = 10

print(array)
print(X)
```

```
[10  2  3  4  5]
[10  2  3  4  5]
```

[70]:
```python
# Array Shape aka Order of Matrix
array = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])
print(array.shape)
```

```
(2, 5)
```

[71]:
```python
# Array Re-Shape
# 1-D To 2-D
array = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9])

newArray = array.reshape(3, 3)
print(newArray)
```

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

[72]:
```
# Array Re-Shape
# 1-D To 3-D
array = np.array([1, 2, 3, 4, 5, 6, 7, 8])

newArray = array.reshape(2, 2, 2)
print(newArray)
```

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

 [[5 6]
  [7 8]]]
```

[73]:
```
# nD Array To 1-D Array
array = np.array([[1, 2, 3], [4, 5, 6]])
print(array)

newArray = array.reshape(-1)
print(newArray)
```

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

[74]:
```
# Iterating Array

# 1-D Array
array = np.array([1, 2, 3, 4, 5])
for x in array:
    print(x)

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

# Another Approach
for x in array:
    for i in x:
        print(i, end="")
```

```
1
2
3
```

```
4
5
[1 2 3 4]
[5 6 7 8]
12345678
```

[75]:
```python
# Iteration Using nditer()
array = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
for x in np.nditer(array):
    print(x, end="")
```

```
12345678
```

[76]:
```python
# Array Join

# 1-D Array
A1 = np.array([1, 2, 3])
A2 = np.array([4, 5, 6])
newArray = np.concatenate((A1, A2))
print(newArray)

# 2-D Array Along Coloums
A1 = np.array([[1, 2], [3, 4]])
A2 = np.array([[5, 6], [7, 8]])
newArray = np.concatenate((A1, A2))
print(newArray)

# 2-D Array Along Rows
A1 = np.array([[1, 2], [3, 4]])
A2 = np.array([[5, 6], [7, 8]])
newArray = np.concatenate((A1, A2), axis=1)
print(newArray)
```

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

[77]:
```python
# 1-D Array Split
array = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9])
newArray = np.array_split(array, 3)
print(newArray)
print(type(newArray))
print(newArray[0])
print(newArray[1])
print(newArray[2])
```

```
[array([1, 2, 3]), array([4, 5, 6]), array([7, 8, 9])]
<class 'list'>
[1 2 3]
[4 5 6]
[7 8 9]
```

[78]:
```
# 2-D Array Split
array = np.array([[1, 2], [3, 4], [5, 6], [7, 8], [9, 10], [11, 12]])
newArray = np.array_split(array, 3)
print(newArray)
```

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

[79]:
```
# Searching Array
array = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 1, 5])
x = np.where(array == 1)
print(x)
```

```
(array([ 0,  9, 11]),)
```

[80]:
```
# Sorting Array

# 1-D Integer
array = np.array([5, 7, 2, 8, 2, 5, 7, 8, 1, 2])
newArray = np.sort(array)
print(newArray)

# 1-D String
array = np.array(["banana", "apple", "yellow", "shadow"])
print(np.sort(array))

# 2-D Integer
array = np.array([[5, 2, 7], [0, 4, 1]])
print(np.sort(array))
```

```
[1 2 2 2 5 5 7 7 8 8]
['apple' 'banana' 'shadow' 'yellow']
[[2 5 7]
 [0 1 4]]
```