NumPy ndarray

- **The array object in NumPy are called ndarray**
- **nd stands for** a **N-dimensions.**
- "N-dimensions" indicates that an array can have any numbers of dimensions.
- A **“Multi-dimensional array”** OR **“N-dimensional array”** OR **“ndarray object”** OR **“NumPy array”** all refer to the same thing: **the ndarray object.**

To find the dimension, shape, size and datatype of a NumPY ndarray

- **ndarray.ndim** - the number of dimensions, of the array.
- **ndarray.size** - the total number of elements of the array. This is the product of the elements of the array’s shape.
- **ndarray.shape** - displays a tuple that indicate the number of elements stored along each dimension of the array (OR a tuple indicating the size of each dimension)
- **ndarray.dtype** - an object describing the data type of the array

```
#Create one-dimensional array
import numpy as np

data1 = [6, 7.5, 8, 0, 1]
a = np.array(data1)

print(a)
print("\n")

print(a.ndim)
print("\n")

print(a.shape)
print("\n")

print(a.size)
print("\n")

print(a.dtype)
print("\n")
```

```
[6.  7.5 8.  0.  1. ]


1
```

```
(5,)


5


float64
```

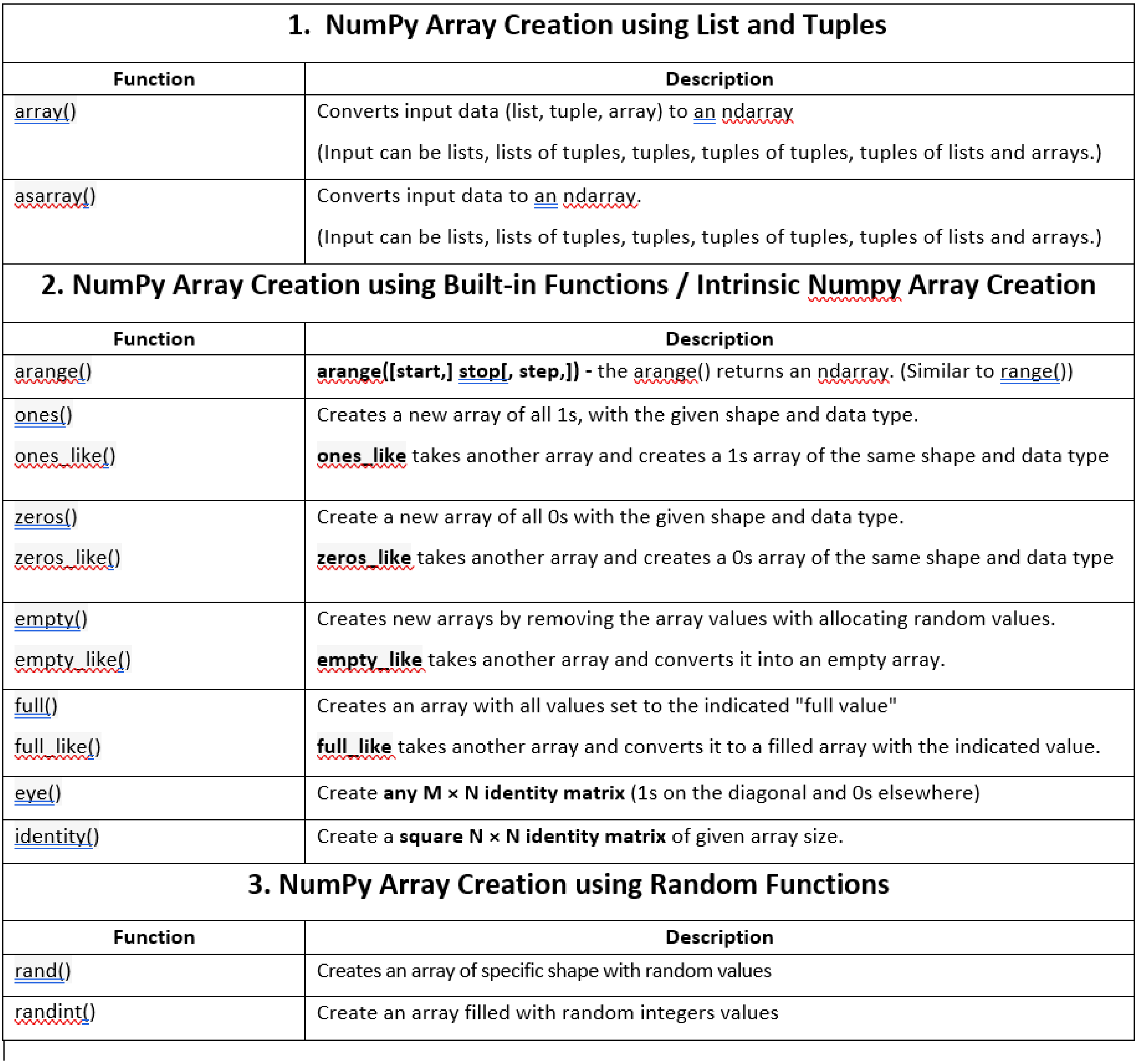
```
#Create two-dimensional array:
import numpy as np

data1 = [[1, 2, 3, 4], [5, 6, 7, 8]]
a = np.array(data1)

print(a)
print("\n")

print(a.ndim)
print("\n")

print(a.shape)
print("\n")

print(a.size)
print("\n")

print(a.dtype)
print("\n")
```

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


2


(2, 4)


8


int64
```

Creating ndarrays

NumPy Arrays can be created using the following three methods:

1. Using Python List and Tuples
2. Using Built-in Functions / **Intrinsic Array Creation**
3. Using Random Functions



1. NumPy Array Creation using List and Tuples

Function	Description
array()	Converts input data (list, tuple, array) to an ndarray (Input can be lists, lists of tuples, tuples, tuples of tuples, tuples of lists and arrays.)
asarray()	Converts input data to an ndarray. (Input can be lists, lists of tuples, tuples, tuples of tuples, tuples of lists and arrays.)

2. NumPy Array Creation using Built-in Functions / Intrinsic Numpy Array Creation

Function	Description
arange()	**arange([start,] stop[, step,])** - the arange() returns an ndarray. (Similar to range())

ones()
ones_like()
	Creates a new array of all 1s, with the given shape and data type.
	ones_like takes another array and creates a 1s array of the same shape and data type

zeros()
zeros_like()
	Create a new array of all 0s with the given shape and data type.
	zeros_like takes another array and creates a 0s array of the same shape and data type

empty()
empty_like()
	Creates new arrays by removing the array values with allocating random values.
	empty_like takes another array and converts it into an empty array.

full()
full_like()
	Creates an array with all values set to the indicated "full value"
	full_like takes another array and converts it to a filled array with the indicated value.

eye()	Create **any M × N identity matrix** (1s on the diagonal and 0s elsewhere)
identity()	Create a **square N × N identity matrix** of given array size.

3. NumPy Array Creation using Random Functions

Function	Description
rand()	Creates an array of specific shape with random values
randint()	Create an array filled with random integers values

Lets see all these functions for creation of NumPy array.

Functions and their python code:

array()

```
import numpy as np
#Create a 0-D array with value 42
a = np.array(42)
print(a)
print("\n")
```

```
#Create a 1-D array containing the values 1,2,3,4,5:
b = np.array([1, 2, 3, 4, 5])
print(b)
print("\n")

#Create a 2-D array containing two arrays with the values 1,2,3 and 4,5,6:
c = np.array([[1, 2, 3], [4, 5, 6]])
print(c)
print("\n")

#Create a 3-D array with two 2-D arrays, both containing two arrays with the values 1,2,3 and 4,5,6:
d = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])
print(d)
print("\n")

#Create an array with 5 dimensions and verify that it has 5 dimensions:
e = np.array([1, 2, 3, 4], ndmin=5)
print(e)
print("\n")
```

```
42


[1 2 3 4 5]


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


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

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


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

arange()

```
print(np.arange(4, 10))
print("\n")

print(np.arange(4, 20, 3))
print("\n")

print(np.arange(4).reshape(2, 2))
print("\n")
```

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

```
[ 4  7 10 13 16 19]


[[0 1]
 [2 3]]
```

ones()

```
b = np.ones(2)
print(b)
print("\n")

a = np.ones([2, 2])
print(a)
print("\n")

c = np.ones([2, 1, 3])
print(c)
print("\n")
```

```
[1. 1.]


[[1. 1.]
 [1. 1.]]


[[[1. 1. 1.]]

 [[1. 1. 1.]]]
```

ones_like()

```
array = np.arange(10)
print(array)
print("\n")

b = np.ones_like(array)
print(b)
print("\n")

array = np.arange(8)
c = np.ones_like(array)
print(c)
```

```
[0 1 2 3 4 5 6 7 8 9]
```

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


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

zeros()

```
b = np.zeros(2)
print(b)
print("\n")

a = np.zeros([2, 2])
print(a)
print("\n")

c = np.zeros([2, 1, 3])
print(c)
print("\n")
```

```
[0. 0.]


[[0. 0.]
 [0. 0.]]


[[[0. 0. 0.]]

 [[0. 0. 0.]]]
```

zeros_like()

```
array = np.arange(10)
print(array)
print("\n")

b = np.zeros_like(array)
print(b)
print("\n")

array = np.arange(8)
c = np.zeros_like(array)
print(c)
print("\n")
```

```
[0 1 2 3 4 5 6 7 8 9]
```

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


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

full()

```
a = np.full([2, 2], 67)
print(a)
print("\n")

c = np.full([3, 3], 10.1)
print( c)
```

```
[[67 67]
 [67 67]]


[[10.1 10.1 10.1]
 [10.1 10.1 10.1]
 [10.1 10.1 10.1]]
```

full_like()

```
import numpy as np

x = np.arange(10, dtype = int).reshape(2, 5)
print( x)
print("\n")

# using full_like
print(np.full_like(x, 10.0))
print("\n")

y = np.arange(10, dtype = float).reshape(2, 5)
print(y)
print("\n")

# using full_like
print(np.full_like(y, 0.01))
```

```
[[0 1 2 3 4]
 [5 6 7 8 9]]


[[10 10 10 10 10]
 [10 10 10 10 10]]
```

```
[[0. 1. 2. 3. 4.]
 [5. 6. 7. 8. 9.]]


[[0.01 0.01 0.01 0.01 0.01]
 [0.01 0.01 0.01 0.01 0.01]]
```

empty()

```
b = np.empty(2, dtype = int)
print(b)
print("\n")

a = np.empty([2, 2])
print(a)
print("\n")

c = np.empty([3, 3])
print(c)
print("\n")
```

```
[0 0]


[[0. 0.]
 [0. 0.]]


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

empty_like()

```
import numpy as np

x = np.arange(10).reshape(2, 5)
print( x)
print("\n")

b = np.empty_like(x)
print(b)
print("\n")
```

```
[[0 1 2 3 4]
 [5 6 7 8 9]]
```

```
[[4576918229304087675 4576918229304087675 4576918229304087675
  4576918229304087675 4576918229304087675]
 [4576918229304087675 4576918229304087675 4576918229304087675
  4576918229304087675 4576918229304087675]]
```

eye()

```
import numpy as np

# 2x2 matrix with 1's on main diagonal
b = np.eye(2)
print( b)
print("\n")

# matrix with R=4 C=5 and 1 on diagonal below main diagonal
a = np.eye(4, 5, k = +2)
print(a)
```

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


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

identity()

```
import numpy as np

# 2x2 matrix with 1's on main diagonal
b = np.identity(2, dtype = float)
print(b)
print("\n")

a = np.identity(4)
print(a)
print("\n")
```

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


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

rand()

```
a = np.random.rand(7)

print(a)
print("\n")

array = np.random.rand(5)
print(array)
print("\n")


array = np.random.rand(2, 2 ,2)
print(array)
print("\n")
```

```
[0.00530639 0.47557389 0.54878056 0.11147181 0.31478528 0.46778881
 0.1809997 ]


[0.06871433 0.01006444 0.52177915 0.78385804 0.47787987]


[[[0.16923368 0.07862075]
  [0.55999678 0.45084972]]

 [[0.69425156 0.49149959]
  [0.58950713 0.95539048]]]
```

✓ 0s completed at 13:21 ● ×