

Numpy 1. Array Creation Functions

```
In [1]: import numpy as np
```

```
In [2]: a=np.array([2,3,4,5])  
print("array a:" ,a)
```

```
array a: [2 3 4 5]
```

```
In [4]: b=np.arange(0,10,2)  
print("array b:",b)
```

```
array b: [0 2 4 6 8]
```

```
In [5]: c=np.zeros((2,3),)  
print("arrays c:\n",c)
```

```
arrays c:  
[[0. 0. 0.]  
 [0. 0. 0.]]
```

```
In [6]: c=np.zeros((2,3),dtype=int)  
print("arrays c:\n",c)
```

```
arrays c:  
[[0 0 0]  
 [0 0 0]]
```

```
In [7]: e=np.ones((2,3),)  
print("arrays e:\n",e)
```

```
arrays e:  
[[1. 1. 1.]  
 [1. 1. 1.]]
```

```
In [8]: e=np.ones((2,3),dtype=int)  
print("arrays e:\n",e)
```

```
arrays e:  
[[1 1 1]  
 [1 1 1]]
```

```
In [9]: f=np.eye(4)  
print("arrays f:\n" ,f)
```

```
arrays f:  
[[1. 0. 0. 0.]  
 [0. 1. 0. 0.]  
 [0. 0. 1. 0.]  
 [0. 0. 0. 1.]]
```

2.Array Manipulation Functions

```
In [10]: a1=np.array([1,2,3,4])  
reshaped= np.reshape(a1,(4,1))  
print("reshaped array:",reshaped)
```

```
reshaped array: [[1]
[2]
[3]
[4]]
```

```
In [11]: a1=np.array([1,2,3,4])#conver array into given shape
reshaped= np.reshape(a1,(2,2))
print("reshaped array:",reshaped)
```

```
reshaped array: [[1 2]
[3 4]]
```

```
In [13]: b1=np.array([[1,2],[3,4],[4,5],[5,6]])#cnvert in to 1darray(1enear)
flattened=np.ravel(b1)
print("flattened array:",flattened)
```

```
flattened array: [1 2 3 4 4 5 5 6]
```

```
In [18]: c1=np.array([[1,2],[3,4]])#convert rows into columns and columns into rows
transposed=np.transpose(c1)
print("transposed array:\n",transposed)
```

```
transposed array:
[[1 3]
[2 4]]
```

```
In [19]: a2=np.array([1,2])#convert array into vertical
b2=np.array([4,5])
stacked=np.stack((a2,b2))
print("stacked array:\n",stacked)
```

```
stacked array:
[[1 2]
[4 5]]
```

3.MATHEMATICAL FUNCTIONS

```
In [23]: g=np.array([1,2,3])
added=np.add(g,2)
print("added 2 to g:",added)
```

```
added 2 to g: [3 4 5]
```

```
In [28]: g=np.array([1,2,3])
added=np.add(g,5)
print("added 2 to g:",added)
```

```
added 2 to g: [6 7 8]
```

```
In [30]: squared =np.power(g,2)
print("squared g:",squared)
```

```
squared g: [1 4 9]
```

```
In [31]: sqrt_val=np.sqrt(g)
print("square root of g:",sqrt_val)
```

```
square root of g: [1.          1.41421356 1.73205081]
```

```
In [32]: print(a)
```

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

```
In [33]: print(a1)
```

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

```
In [35]: a3=np.array([1,2,3,4])
dot_product=np.dot(a,a1)
print("dot product of a and a1:",dot_product)
```

```
dot product of a and a1: 40
```

4.STASTICAL FUNCTIONS

```
In [36]: s=np.array([1,3,4,5,6,])
mean=np.mean(s)
print("mean of array s:",mean)
```

```
mean of array s: 3.8
```

```
In [38]: std_val=np.std(s)
print("std_value of s:",std_val)
```

```
std_value of s: 1.7204650534085253
```

```
In [40]: min=np.min(s)
print("minimum of s:",min)
```

```
minimum of s: 1
```

5.LINEAR ALGEBRA FUNCTIONS

```
In [41]: #create a matrix
matrix=np.array([[2,3],[5,6]])
```

6.Random Sampling Functions

```
In [42]: #generate random values between 0 and 1
random_vals=np.random.rand(3)
print("random values:",random_vals)
```

```
random values: [0.54917191 0.42033546 0.06706985]
```

```
In [44]: np.random.seed(0)#set seed for reproducibility
#generate random values between 0 and 1
random_vals=np.random.rand(3)
print("random values:",random_vals)
```

```
random values: [0.5488135 0.71518937 0.60276338]
```

```
In [46]: #generate random values between 0 and 1
random_ints=np.random.randint(0,10,size=5)
print("random integers:",random_ints)
```

```
random integers: [3 7 9 3 5]
```

```
In [47]: np.random.seed(0)
#generate random values between 0 and 1
```

```
random_ints=np.random.randint(0,10,size=5)
print("random integers:",random_ints)
```

random integers: [5 0 3 3 7]

7.Boolean And Logical Functions

```
In [51]: logical_test=np.array([True,False,True])
all_true=np.all(logical_test)
print("all elements True:",all_true)
```

all elements True: False

```
In [52]: logical_test=np.array([True,True,True])
all_true=np.all(logical_test)
print("all elements True:",all_true)
```

all elements True: True

```
In [55]: logical_test=np.array([True,False,True])
any_true=np.any(logical_test)
print("any elements True:",any_true)
```

any elements True: False

```
In [56]: logical_test=np.array([False,False,False])
all_false=np.all(logical_test)
print("all elements False:",all_false)
```

all elements False: False

8.Set Operations

```
In [58]: set_a=np.array([1,2,3,4])
set_b=np.array([3,4,5,6])
intersection=np.intersect1d(set_a,set_b)
print("intersection of a and b:",intersection)
```

intersection of a and b: [3 4]

```
In [59]: union=np.union1d(set_a,set_b)
print("union of a and b:",union)
```

union of a and b: [1 2 3 4 5 6]

9.Array Attributes

```
In [60]: a=np.array([1,2,3])
shape=a.shape
size=a.size
dimensions=a.ndim
dtype=a.dtype
```

```
In [64]: print("shape of a:",shape)
print("size of a:",size)
print("number of dimensions of a:",dimensions)
print("data type of a:",dtype)
```

shape of a: (3,)

size of a: 3

number of dimensions of a: 1

data type of a: int64

10.Other Functions

```
In [67]: #create a copy of an array  
a=np.array([1,2,3])  
copied_array=np.copy(a)  
print("copied array:",copied_array)
```

copied array: [1 2 3]

```
In [72]: #size in bytes of an array  
array_size_in_bytes = a.nbytes  
print("size of a in bytes:",array_size_in_bytes)
```

size of a in bytes: 24

```
In [74]: #check if two arrays share memory  
shared=np.shares_memory(a,copied_array)  
print("do a and copied_array share memory:",shared)
```

do a and copied_array share memory: False

```
In [ ]:
```

```
In [ ]:
```