# Numpy

In [1]:

```python
import numpy as np
```

In [2]:

```python
# version

np.__version__
```

Out[2]:

```
'1.19.5'
```

In [6]:

```python
# create an array
arr = np.array([1,2,3,4,5])
print(arr)
```

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

In [4]:

```python
type(arr)
```

Out[4]:

```
numpy.ndarray
```

In [12]:

```python
# arange

arr1 = np.arange(10,20)
print(arr1)
```

```
[10 11 12 13 14 15 16 17 18 19]
```

In [13]:

```python
type(arr1)
```

Out[13]:

```
numpy.ndarray
```

In [15]:

```python
arr
```

Out[15]:

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

In [18]:

```python
# check dimension
arr.ndim
```

Out[18]:

```
1
```

In [20]:

```python
arr_2d = np.array([[1,2,3],[4,5,6]])
arr_2d.ndim
```

Out[20]:

```
2
```

In [21]:

```python
print(arr_2d)
```

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

In [22]:

```
# accessing an element

arr[4]
```

Out[22]:

```
5
```

In [23]:

```
arr[0:2]
```

Out[23]:

```
array([1, 2])
```

In [26]:

```
# accessesing an element in 2d

arr_2d[0]
```

Out[26]:

```
array([1, 2, 3])
```

In [28]:

```
arr_2d[1][1]
```

Out[28]:

```
5
```

In [29]:

```
arr_2d[1,1]
```

Out[29]:

```
5
```

In [33]:

```
# shape : defined number of rows and columns

arr_2d.shape
```

Out[33]:

```
(2, 3)
```

In [34]:

```
arr.shape
```

Out[34]:

```
(5,)
```

In [35]:

```
# astype()

arr2 = arr.astype(float)
print(arr2)
```

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

In [36]:

```
# view()

arr.view()
```

Out[36]:

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

In [37]:

```
arr_2d.view()
```

Out[37]:

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

In [38]:

```
# reshape

a = np.array([1,2,3,4,5,6,7,8,9,10,11,12])

```

In [39]:

```
a.shape
```

Out[39]:

```
(12,)
```

In [40]:

```
new_a = a.reshape(3,4)
print(new_a)
```

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

In [41]:

```
new_a.shape
```

Out[41]:

```
(3, 4)
```

In [51]:

```
new_a.size
```

Out[51]:

```
12
```

In [52]:

```
c = np.array([[1,2,3],[4,5,6],[7,8,9],[10,11,12]])
c.size
```

Out[52]:

```
12
```

In [66]:

```
# concatenate

a1 = np.array([1,2,3])
a2 = np.array([6,7,8])


a12 = np.concatenate((a1,a2))
print(a12)
```

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

In [60]:

```
# stacking

st = np.stack((a1,a2),axis = 0) # (row 1 and row 2 as it is)
print(st)
```

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

In [61]:

```
st1 = np.stack((a1,a2),axis = 1) # (row 1  and row 2 becomes ve
print(st1)
```

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

In [62]:

```
# hstack

st_h = np.hstack((a1,a2))
print(st_h)
```

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

In [67]:

```
# vstack

st_v = np.vstack((a1,a2))
print(st_v)
```

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

In [68]:

```
a12
```

Out[68]:

```
array([1, 2, 3, 6, 7, 8])
```

In [72]:

```
# split
split_arr = np.array_split(a12,2)
print(split_arr)
```

```
[array([1, 2, 3]), array([6, 7, 8])]
```

In [73]:

```
# search in 1D

m = np.array([1,2,3,4,5,6])
```

In [74]:

```
s1 = np.where(m==4)
print(s1)
```

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

In [75]:

```
# seach in 2D
st_v
```

Out[75]:

```
array([[1, 2, 3],
       [6, 7, 8]])
```

In [77]:

```
s2 = np.where(st_v==8)
print(s2)
```

```
(array([1]), array([2]))
```

In [92]:

```
# searchsorted
m = np.array([1,2,3,4,5,6])
np.searchsorted(m,3.5)
```

Out[92]:

```
3
```

In [97]:

```
ar = np.array([1.1,2.3,3.7,4.8,5.0,6.1,7.4])
np.searchsorted(ar,3.8)
```

Out[97]:

```
1
```

In [ ]:

```
1.1,2.3,3.7,3.8,4.8,5.0,6.1,7.4
```

In [90]:

```
# searchsorted
z = np.array([1,5,3,4,2,6])
np.searchsorted(z,5.5)
```

Out[90]:

```
5
```

In [80]:

```
# sort

d = np.array([4,7,2,1])
np.sort(d)
```

Out[80]:

```
array([1, 2, 4, 7])
```

In [101]:

```
n =  np.array([1,1,2,2,2,2,3,3,4,4,5,5,6,7])
np.searchsorted(n,5,side = 'right')
```

Out[101]:

```
12
```

In [104]:

```
# summation
a1 = np.array([1,2,3])
a2 = np.array([4,5,6])

new_arr = np.add(a1,a2)
print(new_arr)
```

```
[5 7 9]
```

In [105]:

```
# product

x = np.prod(a1)
print(x)
```

```
6
```

In [107]:

```
# diff
a2 = np.array([4,5,1])
y = np.diff(a2)
print(y)
```

```
[ 1 -4]
```

In [108]:

```
# trigo

a = np.sin(np.pi/2)
print(a)
```

```
1.0
```

In [109]:

```
b = np.cos(np.pi)
print(b)
```

```
-1.0
```

In [110]:

```
# unique

n =  np.array([1,1,2,2,2,2,3,3,4,4,5,5,6,7])
x = np.unique(n)
print(x)
```

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

In [111]:

```
# sum()
ar1 = np.array([4,5,1])
print(np.sum(ar1))
```

```
10
```

In [112]:

```
# mean()
print(np.mean(ar1))
```

```
3.3333333333333335
```

In [113]:

```
# random.rand() : generates random number array between 0 and 1
np.random.rand(2,4)
```

Out[113]:

```
array([[0.51186171, 0.43815185, 0.74742998, 0.36298212],
       [0.15602846, 0.90626437, 0.90217671, 0.89579538]])
```

In [114]:

```
# random.randn() : generates random number which follows normal
np.random.randn(2,4)
```

Out[114]:

```
array([[-1.35420061, -0.84701426, -0.79765137,  0.65219518],
       [-0.71770291, -2.50008705,  0.27444048, -0.35418246]])
```

In [116]:

```
# random.randint(a,b,c) : generates random integers between two
# a  = minimum, b = maximum, c  = number of elements
np.random.randint(3,7,5) #
```

Out[116]:

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

In [118]:

```
# zeros() : generate array of zeros
print(np.zeros((2,2)))
```

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

In [ ]:

```
# ones : generate array of ones # dtype : float ( by default)
```

In [119]:

```
print(np.ones((3,3)))
```

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

In [126]:

```
print(np.ones((3,3),dtype = int))
```

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

In [122]:

```
# ceil() : return ceil value of element
ar = np.array([1.1,2.3,3.7,4.8,5.0,6.1,7.4,9])
np.ceil(ar)
```

Out[122]:

```
array([2., 3., 4., 5., 5., 7., 8., 9.])
```

In [125]:

```
# floor() : return floor value of element

np.floor(ar)
```

Out[125]:

```
array([1., 2., 3., 4., 5., 6., 7., 9.])
```

In [128]:

```
# ravel(): converts multidimentional array to one dimension
ip = np.array([[1,2],[3,4],[7,8]])
np.ravel(ip)
```

Out[128]:

```
array([1, 2, 3, 4, 7, 8])
```

In [129]:

```
# dot() = returns dot product of two array
z1 = np.array([1,2,3])
z2 = np.array([4,5,6])
np.dot(z1,z2)
```

Out[129]:

```
32
```

In [130]:

```
# multiply()
np.multiply(z1,z2)
```

Out[130]:

```
array([ 4, 10, 18])
```

In [133]:

```
# subtract()
z3 = np.array([10,2,3])
z4 = np.array([4,15,6])
np.subtract(z3,z4)
```

Out[133]:

```
array([  6, -13,  -3])
```

In [134]:

```
# log()  :returns natural log of an array
np.log(z3)
```

Out[134]:

```
array([2.30258509, 0.69314718, 1.09861229])
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: