Numpy

August 9, 2023

1 NUMPY

Previously there was a package was a tool is matlab now a days people are using inside a matlab there was lot of mathematical function is available back then and inside a python there was no such mathematical function is avaialable now keeeping that our mind researcher has decided to create a mathematical package inside a python so that then can perform each and every kind of matrices any array based calculation. at the end of the day system just understand array and matrices function . now keeping that our mind researcher have created a package inside a python is numpy or numerical python whaatever operation you want to performnor array and matrices you are able to perform such kind of mathematical function

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

[2]:
```
l =[1,2,3,4,5] # i have a list
```

[3]:
```
np.array(l) # convert list into array
```

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

[4]:
```
arr = np.array(l)
```

[5]:
```
type(arr) # we check type of array
```

[5]:
```
numpy.ndarray
```

[6]:
```
np.asarray(l) # it is a another package asarray whether i using array or␣
 ↪asarray we get same output
```

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

[7]:
```
# i m trying to create a array i have a list inside a list
np.array([[1,2,3],[2,3,4]])
```

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

[8]:
```
arr1 = np.array([[1,2,3],[2,3,4]])
```

[9]:
```
arr.ndim
```

[9]:
```
1
```

[10]:
```
arr1.ndim # ndim is basically telling you the dimension of array
```

[10]:
```
2
```

[11]:
```
arr1
```

[11]:
```
array([[1, 2, 3],
       [2, 3, 4]])
```

[12]:
```
np.matrix(l) # returning matrix will have two 2dimension at a time
```

[12]:
```
matrix([[1, 2, 3, 4, 5]])
```

[13]:
```
mat = np.matrix(l)
```

[14]:
```
np.asanyarray(l)
```

[14]:
```
array([1, 2, 3, 4, 5])
```

[15]:
```
np.asanyarray(mat) # matrices is a sub function inside array one of the type of␣
↪array is a matrices
```

[15]:
```
matrix([[1, 2, 3, 4, 5]])
```

[16]:
```
a = arr
```

[17]:
```
a
```

[17]:
```
array([1, 2, 3, 4, 5])
```

[18]:
```
arr
```

[18]:
```
array([1, 2, 3, 4, 5])
```

[19]:
```
arr[0] = 100 # reasignment operator
```

[20]:
```
arr # this kind of operation is a solo copy operation we are just trying to␣
↪give a reference
```

[20]:
```
array([100,   2,   3,   4,   5])
```

[21]:
```
a
```

[21]:
```
array([100,   2,   3,   4,   5])
```

[22]:
```
b = np.copy(arr)
```

[23]:
```
b
```

[23]:
```
array([100,   2,   3,   4,   5])
```

[24]:
```
b[0] = 234
```

[25]:
```
b
```

[25]:
```
array([234,   2,   3,   4,   5])
```

[26]:
```
arr # b have changed and arr have not changed it is a deep copy concept
```

[26]:
```
array([100,   2,   3,   4,   5])
```

[27]:
```
np.fromfunction(lambda i,j : i == j , (3,3)) # it simply means that it takes␣
↪function as argument and based of the function and nature of function it is␣
↪going to generate a dataset
```

[27]:
```
array([[ True, False, False],
       [False,  True, False],
       [False, False,  True]])
```

[28]:
```
list (i*i for i in range (5))
```

[28]:
```
[0, 1, 4, 9, 16]
```

[29]:
```
iterable = (i*i for i in range(5))
```

[30]:
```
np.fromiter(iterable,float)
```

[30]:
```
array([ 0.,  1.,  4.,  9., 16.])
```

[31]:
```
np.fromstring('23 45 56', sep = ' ' )
```

[31]:
```
array([23., 45., 56.])
```

[32]:
```
np.fromstring('23,45,56', sep = ',' )
```

[32]:
```
array([23., 45., 56.])
```

[33]:
```
arr
```

[33]:
```
array([100,   2,   3,   4,   5])
```

[34]:
```
arr1
```

[34]:
```
array([[1, 2, 3],
       [2, 3, 4]])
```

[35]:
```
arr.ndim
```

[35]:
```
1
```

[36]:
```
arr1.ndim
```

[36]:
```
2
```

[37]:
```
arr.size
```

[37]:
```
5
```

[38]:
```
arr1.size
```

[38]:
```
6
```

[39]:
```
arr.shape
```

[39]:
```
(5,)
```

[40]:
```
arr1.shape # here we check th3e sahpe aof arr1
```

[40]:
```
(2, 3)
```

[41]:
```
arr1
```

[41]:
```
array([[1, 2, 3],
       [2, 3, 4]])
```

[42]:
```
arr.dtype # here we check the dataype od arr
```

[42]:
```
dtype('int64')
```

[43]:
```
arr1.dtype
```

[43]:
```
dtype('int64')
```

[44]:
```
list(range(5))
```

[44]:
```
[0, 1, 2, 3, 4]
```

[45]:
```
list(range(0,10))
```

[45]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

[46]:
```
# list(range(0.4,10.4)) # it will give an error float object can not treated as integer
```

[47]:
```
np.arange(.4,10.4,0.2) # aranjge function will take float value as an argument
```

[47]:
```
array([ 0.4,  0.6,  0.8,  1. ,  1.2,  1.4,  1.6,  1.8,  2. ,  2.2,  2.4,
        2.6,  2.8,  3. ,  3.2,  3.4,  3.6,  3.8,  4. ,  4.2,  4.4,  4.6,
        4.8,  5. ,  5.2,  5.4,  5.6,  5.8,  6. ,  6.2,  6.4,  6.6,  6.8,
        7. ,  7.2,  7.4,  7.6,  7.8,  8. ,  8.2,  8.4,  8.6,  8.8,  9. ,
        9.2,  9.4,  9.6,  9.8, 10. , 10.2])
```

[48]:
```
np.linspace(1,5,20)
```

[48]:
```
array([1.        , 1.21052632, 1.42105263, 1.63157895, 1.84210526,
       2.05263158, 2.26315789, 2.47368421, 2.68421053, 2.89473684,
       3.10526316, 3.31578947, 3.52631579, 3.73684211, 3.94736842,
       4.15789474, 4.36842105, 4.57894737, 4.78947368, 5.        ])
```

[49]:
```
np.logspace(1,5,10, base = 2)
```

[49]:
```
array([ 2.        ,  2.72158   ,  3.70349885,  5.0396842 ,  6.85795186,
        9.33223232, 12.69920842, 17.28095582, 23.51575188, 32.        ])
```

[50]:
```
np.zeros(5)
```

[50]:
```
array([0., 0., 0., 0., 0.])
```

[51]:
```
np.zeros((3,4))
```

[51]:
```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

[52]:
```
np.zeros((3,4,2))
```

[52]:
```
array([[[0., 0.],
        [0., 0.],
        [0., 0.],
        [0., 0.]],

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

```
       [[0., 0.],
        [0., 0.],
        [0., 0.],
        [0., 0.]]])
```

[53]:
```
np.ones(5)
```

[53]:
```
array([1., 1., 1., 1., 1.])
```

[54]:
```
np.ones((3,4))
```

[54]:
```
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.]])
```

[55]:
```
arr = np.ones((3,4))
```

[56]:
```
arr
```

[56]:
```
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.]])
```

[57]:
```
arr+5
```

[57]:
```
array([[6., 6., 6., 6.],
       [6., 6., 6., 6.],
       [6., 6., 6., 6.]])
```

[58]:
```
np.empty((3,4))
```

[58]:
```
array([[6., 6., 6., 6.],
       [6., 6., 6., 6.],
       [6., 6., 6., 6.]])
```

[59]:
```
np.eye(3)
```

[59]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

[60]:
```
import pandas as pd
```

[61]:
```
pd.DataFrame(arr)
```

[61]:

	0	1	2	3
0	1.0	1.0	1.0	1.0

```
1  1.0  1.0  1.0  1.0
2  1.0  1.0  1.0  1.0
```

[62]:
```
np.random.rand(2,3)
```

[62]:
```
array([[0.65269029, 0.79902212, 0.32431889],
       [0.75056608, 0.92718876, 0.17008752]])
```

[63]:
```
np.random.randn(2,3)
```

[63]:
```
array([[-0.49594285, -1.18042462, -2.64521373],
       [ 0.72577166,  1.60547492,  0.11898726]])
```

[64]:
```
np.random.randint(1,5,(3,4))
```

[64]:
```
array([[1, 2, 4, 1],
       [4, 2, 1, 2],
       [3, 4, 3, 1]])
```

[65]:
```
arr2 = np.random.randint(1,5,(3,4))
```

[66]:
```
arr2.size
```

[66]:
```
12
```

[67]:
```
arr2.shape
```

[67]:
```
(3, 4)
```

[68]:
```
arr2.reshape(6,2) # change the shape of dataset value passes is the multipiler
↪of the
```

[68]:
```
array([[2, 3],
       [4, 1],
       [2, 3],
       [2, 2],
       [2, 4],
       [4, 3]])
```

[69]:
```
arr1 = np.random.randint(1,10,(5,6))
```

[70]:
```
arr1
```

[70]:
```
array([[3, 3, 4, 5, 1, 9],
       [3, 5, 7, 5, 1, 6],
       [5, 4, 1, 5, 8, 8],
       [9, 8, 1, 9, 1, 1],
```

```
       [6, 1, 8, 3, 4, 5]])
```

[71]:
```
arr1>8
```

[71]:
```
array([[False, False, False, False, False,  True],
       [False, False, False, False, False, False],
       [False, False, False, False, False, False],
       [ True, False, False,  True, False, False],
       [False, False, False, False, False, False]])
```

[72]:
```
arr1[arr1>8]
```

[72]:
```
array([9, 9, 9])
```

[73]:
```
arr1
```

[73]:
```
array([[3, 3, 4, 5, 1, 9],
       [3, 5, 7, 5, 1, 6],
       [5, 4, 1, 5, 8, 8],
       [9, 8, 1, 9, 1, 1],
       [6, 1, 8, 3, 4, 5]])
```

[74]:
```
arr1[0]
```

[74]:
```
array([3, 3, 4, 5, 1, 9])
```

[75]:
```
arr1[0,[0,1]]
```

[75]:
```
array([3, 3])
```

[76]:
```
arr1[2:4,[2,3]]
```

[76]:
```
array([[1, 5],
       [1, 9]])
```

[77]:
```
arr1 = np.random.randint(1,3,(3,3))
arr2 = np.random.randint(1,3,(3,3))
```

[78]:
```
arr1
```

[78]:
```
array([[1, 1, 2],
       [1, 1, 1],
       [2, 2, 2]])
```

[79]:
```
arr2
```

[79]:
```
array([[2, 2, 2],
       [1, 2, 1],
       [1, 1, 2]])
```

[80]:
```
arr1+arr2
```

[80]:
```
array([[3, 3, 4],
       [2, 3, 2],
       [3, 3, 4]])
```

[81]:
```
arr1*arr2
```

[81]:
```
array([[2, 2, 4],
       [1, 2, 1],
       [2, 2, 4]])
```

[82]:
```
arr1-arr2
```

[82]:
```
array([[-1, -1,  0],
       [ 0, -1,  0],
       [ 1,  1,  0]])
```

[83]:
```
arr1@arr2
```

[83]:
```
array([[ 5,  6,  7],
       [ 4,  5,  5],
       [ 8, 10, 10]])
```

[84]:
```
arr1/arr2
```

[84]:
```
array([[0.5, 0.5, 1. ],
       [1. , 0.5, 1. ],
       [2. , 2. , 1. ]])
```

[85]:
```
arr1/0
```

```
/tmp/ipykernel_4856/1510032488.py:1: RuntimeWarning: divide by zero encountered
in divide
  arr1/0
```

[85]:
```
array([[inf, inf, inf],
       [inf, inf, inf],
       [inf, inf, inf]])
```

[86]:
```
# numpy broadcasting
```

[87]:
```
arr = np.zeros((3,4))
```

[88]:
```
arr
```

[88]:
```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

[89]:
```
arr+5
```

[89]:
```
array([[5., 5., 5., 5.],
       [5., 5., 5., 5.],
       [5., 5., 5., 5.]])
```

[90]:
```
a = np.array([1,2,3,4,])
```

[91]:
```
arr+a
```

[91]:
```
array([[1., 2., 3., 4.],
       [1., 2., 3., 4.],
       [1., 2., 3., 4.]])
```

[92]:
```
b = np.array([[3,4,5]])
```

[93]:
```
arr+b.T
```

[93]:
```
array([[3., 3., 3., 3.],
       [4., 4., 4., 4.],
       [5., 5., 5., 5.]])
```

[94]:
```
b.T
```

[94]:
```
array([[3],
       [4],
       [5]])
```

[95]:
```
arr1
```

[95]:
```
array([[1, 1, 2],
       [1, 1, 1],
       [2, 2, 2]])
```

[96]:
```
np.sqrt(arr1)
```

[96]:
```
array([[1.        , 1.        , 1.41421356],
       [1.        , 1.        , 1.        ],
       [1.41421356, 1.41421356, 1.41421356]])
```

[97]:
```
np.log10(arr1)
```

[97]:
```
array([[0.     , 0.     , 0.30103],
       [0.     , 0.     , 0.     ],
       [0.30103, 0.30103, 0.30103]])
```

[98]:
```
np.exp(arr1)
```

[98]:
```
array([[2.71828183, 2.71828183, 7.3890561 ],
       [2.71828183, 2.71828183, 2.71828183],
       [7.3890561 , 7.3890561 , 7.3890561 ]])
```

[99]:
```
np.min(arr1)
```

[99]:
```
1
```

[100]:
```
np.max(arr1)
```

[100]:
```
2
```

[101]:
```
# Numpy - Array Manipulation
```

[102]:
```
np.random.randint(1,10,(4,4))
```

[102]:
```
array([[8, 3, 3, 3],
       [5, 5, 5, 6],
       [3, 5, 3, 2],
       [4, 4, 5, 5]])
```

[103]:
```
arr3 = np.random.randint(1,10,(4,4))
```

[104]:
```
arr3
```

[104]:
```
array([[2, 5, 5, 8],
       [2, 9, 9, 1],
       [6, 1, 5, 9],
       [4, 3, 5, 1]])
```

[105]:
```
# reshape the array
arr3.reshape(8,2)
```

[105]:
```
array([[2, 5],
       [5, 8],
       [2, 9],
       [9, 1],
       [6, 1],
       [5, 9],
       [4, 3],
       [5, 1]])
```

[106]:
```
arr3.T
```

[106]:
```
array([[2, 2, 6, 4],
       [5, 9, 1, 3],
       [5, 9, 5, 5],
       [8, 1, 9, 1]])
```

[107]:
```
arr3.flatten() # array in a single one dimension
```

[107]:
```
array([2, 5, 5, 8, 2, 9, 9, 1, 6, 1, 5, 9, 4, 3, 5, 1])
```

[108]:
```
arr3
```

[108]:
```
array([[2, 5, 5, 8],
       [2, 9, 9, 1],
       [6, 1, 5, 9],
       [4, 3, 5, 1]])
```

[109]:
```
# here we expend the dimension of the array
np.expand_dims(arr3,axis=1)
```

[109]:
```
array([[[2, 5, 5, 8]],

       [[2, 9, 9, 1]],

       [[6, 1, 5, 9]],

       [[4, 3, 5, 1]]])
```

[110]:
```
data = np.array([[1],[2],[3]])
```

[111]:
```
# squeezes is the function that changes the dimensions
np.squeeze(data)
```

[111]:
```
array([1, 2, 3])
```

[112]:
```
np.repeat(data,2)
```

[112]:
```
array([1, 1, 2, 2, 3, 3])
```

[113]:
```
np.roll(data,2) # roll is rotate the datset
```

[113]:
```
array([[2],
       [3],
       [1]])
```

[114]:
```
np.diag(np.array([1,2,3,4]))
```

[114]:
```
array([[1, 0, 0, 0],
       [0, 2, 0, 0],
       [0, 0, 3, 0],
       [0, 0, 0, 4]])
```

[115]:
```
# Numpy - Binary operator
```

[116]:
```
arr1 = np.random.randint(1,10,(3,4))
arr2 = np.random.randint(1,10,(3,4))
```

[117]:
```
arr1
```

[117]:
```
array([[2, 2, 7, 3],
       [7, 3, 6, 7],
       [8, 9, 4, 6]])
```

[118]:
```
arr2
```

[118]:
```
array([[6, 1, 1, 6],
       [5, 6, 3, 1],
       [8, 4, 6, 9]])
```

[119]:
```
arr1*arr2
```

[119]:
```
array([[12,  2,  7, 18],
       [35, 18, 18,  7],
       [64, 36, 24, 54]])
```

[120]:
```
arr1-arr2
```

[120]:
```
array([[-4,  1,  6, -3],
       [ 2, -3,  3,  6],
       [ 0,  5, -2, -3]])
```

[121]:
```
arr1>arr2
```

[121]:
```
array([[False,  True,  True, False],
       [ True, False,  True,  True],
       [False,  True, False, False]])
```

[122]:
```
# Numpy string functions
```

[123]:
```
arr = np.array(["Priyanshu","Gupta"])
```

[124]:
```
arr
```

[124]:
```
array(['Priyanshu', 'Gupta'], dtype='<U9')
```

[125]:
```
np.char.upper(arr)
```

[125]:
```
array(['PRIYANSHU', 'GUPTA'], dtype='<U9')
```

[126]:
```
np.char.capitalize(arr)
```

[126]:
```
array(['Priyanshu', 'Gupta'], dtype='<U9')
```

[127]:
```
np.char.title(arr)
```

[127]:
```
array(['Priyanshu', 'Gupta'], dtype='<U9')
```

[128]:
```
# Numpy - mathematical function
```

[129]:
```
arr1
```

[129]:
```
array([[2, 2, 7, 3],
       [7, 3, 6, 7],
       [8, 9, 4, 6]])
```

[130]:
```
np.sin(arr1)
```

[130]:
```
array([[ 0.90929743,  0.90929743,  0.6569866 ,  0.14112001],
       [ 0.6569866 ,  0.14112001, -0.2794155 ,  0.6569866 ],
       [ 0.98935825,  0.41211849, -0.7568025 , -0.2794155 ]])
```

[131]:
```
np.cos(arr1)
```

[131]:
```
array([[-0.41614684, -0.41614684,  0.75390225, -0.9899925 ],
       [ 0.75390225, -0.9899925 ,  0.96017029,  0.75390225],
       [-0.14550003, -0.91113026, -0.65364362,  0.96017029]])
```

[132]:
```
np.tan(arr1)
```

[132]:
```
array([[-2.18503986, -2.18503986,  0.87144798, -0.14254654],
       [ 0.87144798, -0.14254654, -0.29100619,  0.87144798],
       [-6.79971146, -0.45231566,  1.15782128, -0.29100619]])
```

[133]:
```
np.log10(arr1)
```

[133]:
```
array([[0.30103   , 0.30103   , 0.84509804, 0.47712125],
       [0.84509804, 0.47712125, 0.77815125, 0.84509804],
       [0.90308999, 0.95424251, 0.60205999, 0.77815125]])
```

[134]:
```
np.exp(arr1)
```

[134]:
```
array([[7.38905610e+00, 7.38905610e+00, 1.09663316e+03, 2.00855369e+01],
       [1.09663316e+03, 2.00855369e+01, 4.03428793e+02, 1.09663316e+03],
       [2.98095799e+03, 8.10308393e+03, 5.45981500e+01, 4.03428793e+02]])
```

[135]:
```
np.power(arr1,2)
```

[135]:
```
array([[ 4,  4, 49,  9],
       [49,  9, 36, 49],
       [64, 81, 16, 36]])
```

[136]:
```
np.mean(arr1)
```

[136]:
```
5.333333333333333
```

[137]:
```
np.median(arr1)
```

[137]:
```
6.0
```

[138]:
```
np.std(arr1)
```

[138]:
```
2.321398046197353
```

[139]:
```
np.var(arr1)
```

[139]:
```
5.388888888888888
```

[140]:
```
np.min(arr1)
```

[140]:
```
2
```

[141]:
```
np.max(arr1)
```

[141]:
```
9
```

[142]:
```
# Numpy - Arithematic operations
```

[143]:
```
arr1
```

[143]:
```
array([[2, 2, 7, 3],
       [7, 3, 6, 7],
       [8, 9, 4, 6]])
```

[144]:
```
arr2
```

[144]:
```
array([[6, 1, 1, 6],
       [5, 6, 3, 1],
       [8, 4, 6, 9]])
```

[145]:
```
arr1-arr2
```

[145]:
```
array([[-4,  1,  6, -3],
       [ 2, -3,  3,  6],
       [ 0,  5, -2, -3]])
```

[146]:
```
np.subtract(arr1,arr2)
```

[146]:
```
array([[-4,  1,  6, -3],
       [ 2, -3,  3,  6],
       [ 0,  5, -2, -3]])
```

[147]:
```
np.multiply(arr1,arr2)
```

[147]:
```
array([[12,  2,  7, 18],
       [35, 18, 18,  7],
       [64, 36, 24, 54]])
```

[148]:
```
np.power(arr1,arr2)
```

[148]:
```
array([[      64,        2,        7,      729],
       [   16807,      729,      216,        7],
       [16777216,     6561,     4096, 10077696]])
```

[149]:
```
np.sqrt(arr1)
```

[149]:
```
array([[1.41421356, 1.41421356, 2.64575131, 1.73205081],
       [2.64575131, 1.73205081, 2.44948974, 2.64575131],
       [2.82842712, 3.        , 2.        , 2.44948974]])
```

[150]:
```
np.std(arr1)
```

[150]:
```
2.321398046197353
```

[151]:
```
np.median(arr1)
```

[151]:
```
6.0
```

[152]:
```
# sort search & counting function
```

[153]:
```
arr = np.array([2,3,4,5,6])
```

[154]:
```
arr
```

[154]:
```
array([2, 3, 4, 5, 6])
```

[155]:
```
np.sort(arr)
```

[155]:
```
array([2, 3, 4, 5, 6])
```

[156]:
```
np.searchsorted(arr,6)
```

[156]:
```
4
```

[157]:
```
arr1 = np.array([0,324,645,65,6,6,0,0,0,234])
```

[158]:
```
np.count_nonzero(arr1)
```

[158]:
```
6
```

[160]:
```
np.where(arr>0) # where give the indexes which is greater than zero
```

[160]:
```
(array([0, 1, 2, 3, 4]),)
```

[164]:
```
np.extract(arr1>2,arr1)
```

[164]:
```
array([324, 645,  65,   6,   6, 234])
```

[165]:
```
# Numpy - Byte swapping
```

[166]:
```
arr1
```

[166]:
```
array([  0, 324, 645,  65,   6,   6,   0,   0,   0, 234])
```

[167]:
```
arr.byteswap()
```

[167]:
```
array([144115188075855872, 216172782113783808, 288230376151711744,
       360287970189639680, 432345564227567616])
```

[168]:
```
arr
```

[168]:
```
array([2, 3, 4, 5, 6])
```

[169]:
```
arr.byteswap(True)
```

[169]:
```
array([144115188075855872, 216172782113783808, 288230376151711744,
       360287970189639680, 432345564227567616])
```

[170]:
```
# Numpy - Copies & views
```

[171]:
```
arr1
```

[171]:
```
array([  0, 324, 645,  65,   6,   6,   0,   0,   0, 234])
```

[172]:
```
a = np.copy(arr1)
```

[173]:
```
b = arr1.view()
```

[174]:
```
b
```

[174]:
```
array([  0, 324, 645,  65,   6,   6,   0,   0,   0, 234])
```

[175]:
```
b[0] = 234
```

[176]:
```
b
```

[176]:
```
array([234, 324, 645,  65,   6,   6,   0,   0,   0, 234])
```

[177]:
```
arr1
```

[177]:
```
array([234, 324, 645,  65,   6,   6,   0,   0,   0, 234])
```

[178]:
```
# Numpy - Matrix library
```

[179]:
```
import numpy.matlib  as nm
```

[180]:
```
nm.zeros(5)
```

[180]:
```
matrix([[0., 0., 0., 0., 0.]])
```

[181]:
```
nm.ones((3,4))
```

[181]:
```
matrix([[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]])
```

[182]:
```
nm.eye(4)
```

[182]:
```
matrix([[1., 0., 0., 0.],
        [0., 1., 0., 0.],
        [0., 0., 1., 0.],
        [0., 0., 0., 1.]])
```

[185]:
```
arr1 = np.random.randint([[2,3],[4,5]])
arr2 = np.random.randint([[5,3],[2,5]])
```

[186]:
```
arr1
```

[186]:
```
array([[0, 1],
       [1, 0]])
```

[187]:
```
arr2
```

[187]:
```
array([[0, 2],
       [1, 2]])
```

[188]:
```
np.dot(arr1,arr2)
```

[188]:
```
array([[1, 2],
       [0, 2]])
```

[189]:
```
arr1@arr2
```

[189]:
```
array([[1, 2],
       [0, 2]])
```