numpy-data-science-2

April 23, 2024

1 NumPy Exercises

Now that we have learned about Numpy let's test your knowledge. We'll start off with a New simple tasks, and then you'll be asked some more complicated questions

2 Import NumPy as np

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

3 Create an array of 10 zeroes

[2]:
```
np.zeros(10)
```

[2]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

4 create an array of 10 ones

[6]:
```
np.ones(10)
```

[6]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

5 create an array of 10 fives.

[7]:
```
np.ones(10)*5
```

[7]:
```
array([5., 5., 5., 5., 5., 5., 5., 5., 5., 5.])
```

[8]:
```
#create an array of the integers from 10 to 50
```

[7]:
```
np.arange(10,50)
```

[7]:
```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
       27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43,
       44, 45, 46, 47, 48, 49])
```

[10]:
```
#create an array of all the even integers from 10 to 50
```

[11]:
```
np.arange(10,50,2)
```

[11]:
```
array([10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
       44, 46, 48])
```

[12]:
```
#create an 3x3 matrix with values ranging from 0 to 8
```

[8]:
```
np.arange(0,9).reshape(3,3)
```

[8]:
```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

[15]:
```
#create a 3x3 identity matrix
```

[16]:
```
np.eye(3)
```

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

[17]:
```
#use Numpy to generate a random integer between 0 and 1
```

[3]:
```
import numpy as np
np.random.rand(1)
```

[3]:
```
array([0.68026481])
```

[23]:
```
#use Numpy to generate an array of 23 random numbers sampled from a standard␣
↪normal distribution
```

[3]:
```
np.random.randn(23)
```

[3]:
```
array([-0.70223965, -0.96252531,  0.07290015,  0.04525854, -0.19082715,
        0.19557492, -0.52334979,  0.36468822, -1.56438197, -0.46370973,
        1.6421112 ,  0.75887653,  0.1384222 , -0.50416373, -0.9254727 ,
        2.32683447,  0.02791543, -1.54490699,  0.2058696 ,  0.36333381,
        0.65822785,  1.42645253, -0.26923351])
```

[25]:
```
#Create the following matrix
```

[41]:
```
np.arange(1,101).reshape(10,10)/100
```

[41]:
```
array([[0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1 ],
       [0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 ],
```

```
       [0.21, 0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 ],
       [0.31, 0.32, 0.33, 0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 ],
       [0.41, 0.42, 0.43, 0.44, 0.45, 0.46, 0.47, 0.48, 0.49, 0.5 ],
       [0.51, 0.52, 0.53, 0.54, 0.55, 0.56, 0.57, 0.58, 0.59, 0.6 ],
       [0.61, 0.62, 0.63, 0.64, 0.65, 0.66, 0.67, 0.68, 0.69, 0.7 ],
       [0.71, 0.72, 0.73, 0.74, 0.75, 0.76, 0.77, 0.78, 0.79, 0.8 ],
       [0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87, 0.88, 0.89, 0.9 ],
       [0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99, 1.  ]])
```

[42]:
```
#create an array of 20 linearly spaced points between 0 and 1
```

[6]:
```
np.linspace(0,1,20)
```

[6]:
```
array([0.        , 0.05263158, 0.10526316, 0.15789474, 0.21052632,
       0.26315789, 0.31578947, 0.36842105, 0.42105263, 0.47368421,
       0.52631579, 0.57894737, 0.63157895, 0.68421053, 0.73684211,
       0.78947368, 0.84210526, 0.89473684, 0.94736842, 1.        ])
```

[44]:
```
#Numpy indexing and selection
```

[45]:
```
#Now you will be given a new matrices and be asked to replicate the resulting␣
↪matrix outputs
```

[8]:
```
a=np.arange(1,26).reshape(5,5)
a
```

[8]:
```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

[48]:
```
#write code here that reproduces the output of the cell function
#be careful not to run the cell below, otherwise you won't
#be able to see the output any more
```

[34]:
```
a[1:,2:5]
```

[34]:
```
array([[ 8,  9, 10],
       [13, 14, 15],
       [18, 19, 20],
       [23, 24, 25]])
```

[13]:
```
a[2:,1:]
```

[13]:
```
array([[12, 13, 14, 15],
       [17, 18, 19, 20],
```

```
       [22, 23, 24, 25]])
```

[69]:
```
a[3,4]
```

[69]:
```
20
```

[14]:
```
a[3:4,4]
```

[14]:
```
array([20])
```

[15]:
```
a[:3,1].reshape(3,1)
```

[15]:
```
array([[ 2],
       [ 7],
       [12]])
```

[16]:
```
a[4:]
```

[16]:
```
array([[21, 22, 23, 24, 25]])
```

[17]:
```
a[3:]
```

[17]:
```
array([[16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

[87]:
```
#Now do the following
```

[88]:
```
#get the sum of all the values in matrix
```

[18]:
```
a.sum()
```

[18]:
```
325
```

[91]:
```
#get the standard deviation of value in the matrix
```

[19]:
```
a.std()
```

[19]:
```
7.211102550927978
```

[93]:
```
#get the sum of all the columns in the matrix
```

[20]:
```
sum(a)
```

[20]:
```
array([55, 60, 65, 70, 75])
```

[21]:
```
a[0].sum(),a[1].sum(),a[2].sum(),a[3].sum(),a[4].sum()
```

[21]:
```
(15, 40, 65, 90, 115)
```

[12]:
```
a.max(axis=0)
```

[12]:
```
array([21, 22, 23, 24, 25])
```

[99]:
```
a.sum(axis=1)
```

[99]:
```
array([ 15,  40,  65,  90, 115])
```

6 Numpy Exercise 2

[101]:
```
#Create 1d,2d and 3d numpy array
```

[23]:
```
b=np.arange(1,17).reshape(4,4)
b
```

[23]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12],
       [13, 14, 15, 16]])
```

[26]:
```
c=np.arange(1,17)
c
```

[26]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16])
```

[27]:
```
import numpy as np
d=np.arange(1,17).reshape(4,2,2)
d
```

[27]:
```
array([[[ 1,  2],
        [ 3,  4]],

       [[ 5,  6],
        [ 7,  8]],

       [[ 9, 10],
        [11, 12]],

       [[13, 14],
        [15, 16]]])
```

[116]:
```
#print data type of array
```

[28]:
```
b.dtype
c.dtype
d.dtype
```

[28]:
```
dtype('int32')
```

[129]:
```
#print shape and dimension of all the three different array
```

[29]:
```
b.shape
```

[29]:
```
(4, 4)
```

[30]:
```
d.shape
```

[30]:
```
(4, 2, 2)
```

[31]:
```
c.shape
```

[31]:
```
(16,)
```

[145]:
```
#create another 1d array of 12 elements filled with zeros
```

[32]:
```
np.zeros(12)
```

[32]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

[143]:
```
#create another 2d array of elements filled with ones
```

[33]:
```
np.ones((3,4))
```

[33]:
```
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.]])
```

[147]:
```
#create a 1d array known as Maths marks out of 100 having 20 elements
```

[39]:
```
marks=np.random.randint(0,101,20)
marks
```

[39]:
```
array([18, 77, 14, 60, 13, 26, 79, 58, 63, 44, 99, 85, 98, 80, 60, 34,  4,
       57, 77, 59])
```

[35]:
```
d.ndim
```

[35]:
```
3
```

[36]:
```
#Sort the above array in assending and decending order
```

[44]:
```
asc=np.sort(marks)
asc
```

[44]:
```
array([ 4, 13, 14, 18, 26, 34, 44, 57, 58, 59, 60, 60, 63, 77, 77, 79, 80,
       85, 98, 99])
```

[37]:
```
#Create a copy of sorted array to another variable by deep copying
```

[45]:
```
dsc=np.flip(asc)
dsc
```

[45]:
```
array([99, 98, 85, 80, 79, 77, 77, 63, 60, 60, 59, 58, 57, 44, 34, 26, 18,
       14, 13,  4])
```

[60]:
```
ac=asc.copy()
ac
```

[60]:
```
array([ 4, 13, 14, 18, 26, 34, 44, 57, 58, 59, 60, 60, 63, 77, 77, 79, 80,
       85, 98, 99])
```

[43]:
```
#Create a 2d array of 5x5 which is an identity matrix and then convert to 1-d␣
↪array
```

[61]:
```
e=np.arange(1,26).reshape(5,5)
e
f=e.reshape(25)
f
```

[61]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24, 25])
```

[62]:
```
np.eye(5).flatten()
```

[62]:
```
array([1., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0.,
       0., 1., 0., 0., 0., 0., 0., 1.])
```

[63]:
```
np.eye(5).ravel()
```

[63]:
```
array([1., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0.,
       0., 1., 0., 0., 0., 0., 0., 1.])
```

[67]:
```
np.eye(5).reshape(25,)
```

[67]:
```
array([1., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0.,
       0., 1., 0., 0., 0., 0., 0., 1.])
```

[73]:
```
#Create a 2d array of 3x3.Add one more row to it.(vstack,hstack)
```

[15]:
```
mat=np.arange(0,9).reshape(3,3)
mat
```

[15]:
```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

[30]:
```
f=np.append(mat,[[9,10,11]],axis=0)
f
```

[30]:
```
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])
```

[47]:
```
#Split above array created vertically and horizontally(hsplit, vsplit)
```

[87]:
```
np.split(f,4)
```

[87]:
```
[array([[0, 1, 2]]),
 array([[3, 4, 5]]),
 array([[6, 7, 8]]),
 array([[ 9, 10, 11]])]
```

[93]:
```
np.hsplit(f,3)
```

[93]:
```
[array([[0],
        [3],
        [6],
        [9]]),
 array([[ 1],
        [ 4],
        [ 7],
        [10]]),
 array([[ 2],
        [ 5],
        [ 8],
        [11]])]
```

[94]:
```
np.vsplit(f,4)
```

[94]:
```
[array([[0, 1, 2]]),
 array([[3, 4, 5]]),
 array([[6, 7, 8]]),
 array([[ 9, 10, 11]])]
```

[51]:
```
#Create 4x4 identity matrix.Add 4 to each element.
```

[99]:
```
g=np.eye(4)
g
```

[99]:
```
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]])
```

[100]:
```
g+4
```

[100]:
```
array([[5., 4., 4., 4.],
       [4., 5., 4., 4.],
       [4., 4., 5., 4.],
       [4., 4., 4., 5.]])
```

[52]:
```
#Create 1d array. Initialize with floating point values. Display its ceil,␣
↪floor and round(with)
```

[6]:
```
h=np.random.uniform(1,9,10)
h
```

[6]:
```
array([6.02388616, 2.25684908, 6.63432705, 3.50580865, 8.12549572,
       1.3620505 , 8.83805567, 7.18137196, 8.05614724, 8.13031558])
```

[7]:
```
np.ceil(h)
```

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

[8]:
```
np.floor(h)
```

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

[10]:
```
np.round(h,3)
```

[10]:
```
array([6.024, 2.257, 6.634, 3.506, 8.125, 1.362, 8.838, 7.181, 8.056,
       8.13 ])
```

[53]:
```
#Create 5x5 matrix. Display min, max , variance and standard deviation column␣
↪wise.
```

[1]:
```
import numpy as np
i=np.arange(1,26).reshape(5,5)
i
```

[1]:
```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15],
```

```
       [16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

[13]:
```
i.min(axis=1)
```

[13]:
```
array([ 1,  6, 11, 16, 21])
```

[14]:
```
i.max(axis=1)
```

[14]:
```
array([ 5, 10, 15, 20, 25])
```

[5]:
```
i.std(axis=1)
```

[5]:
```
array([1.41421356, 1.41421356, 1.41421356, 1.41421356, 1.41421356])
```

[6]:
```
i.var(axis=1)
```

[6]:
```
array([2., 2., 2., 2., 2.])
```

[54]:
```
#Create a null vector of size 10 with 6th value filled with 200.
```

[147]:
```
j=np.zeros(10)
j[5]=200
j
```

[147]:
```
array([  0.,   0.,   0.,   0.,   0., 200.,   0.,   0.,   0.,   0.])
```

[130]:
```
#Create a 1d array/vector with values ranging from 100 to 150 in sequence.
```

[137]:
```
k=np.arange(100,151)
k
```

[137]:
```
array([100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112,
       113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125,
       126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138,
       139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150])
```

[101]:
```
#Reverse the above vector.
```

[138]:
```
np.flip(k)
```

[138]:
```
array([150, 149, 148, 147, 146, 145, 144, 143, 142, 141, 140, 139, 138,
       137, 136, 135, 134, 133, 132, 131, 130, 129, 128, 127, 126, 125,
       124, 123, 122, 121, 120, 119, 118, 117, 116, 115, 114, 113, 112,
       111, 110, 109, 108, 107, 106, 105, 104, 103, 102, 101, 100])
```

[102]:
```
#Create a 5x5 matrix with values ranging from 0-24 using some function
```

[139]:
```
np.arange(0,25).reshape(5,5)
```

[139]:
```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24]])
```

[115]:
```
#Find indices of nonzero elements from [1,0,2,0,0,4,0,0]
```

[148]:
```
l=np.array([1,0,2,0,0,4,0,0])
np.nonzero(l)
```

[148]:
```
(array([0, 2, 5], dtype=int64),)
```

[142]:
```
#Create a 10x10 matrix with 1 on the borders and 0 inside
```

[5]:
```
import numpy as np

# Create a 2D matrix filled with nan
matrix = np.ones((8,8))*5

# Set the border values to nan
matrix[:, [0, -1]] = "nan"  # Set the first and last columns to nan
matrix[[0, -1], :] ="nan"# Set the first and last rows to nan

# Display the resulting matrix
print(matrix)
```

```
[[nan nan nan nan nan nan nan nan]
 [nan  5.  5.  5.  5.  5.  5. nan]
 [nan  5.  5.  5.  5.  5.  5. nan]
 [nan  5.  5.  5.  5.  5.  5. nan]
 [nan  5.  5.  5.  5.  5.  5. nan]
 [nan  5.  5.  5.  5.  5.  5. nan]
 [nan  5.  5.  5.  5.  5.  5. nan]
 [nan nan nan nan nan nan nan nan]]
```

[159]:
```
#Create 8x8 matrix and fill with 1 and 0 in checkerbox pattern
```

[171]:
```
n = np.zeros((8, 8), dtype=int)
n[::2, ::2] = 1
n[1::2, 1::2] = 1
n
```

[171]:
```
array([[1, 0, 1, 0, 1, 0, 1, 0],
       [0, 1, 0, 1, 0, 1, 0, 1],
```

```
       [1, 0, 1, 0, 1, 0, 1, 0],
       [0, 1, 0, 1, 0, 1, 0, 1],
       [1, 0, 1, 0, 1, 0, 1, 0],
       [0, 1, 0, 1, 0, 1, 0, 1],
       [1, 0, 1, 0, 1, 0, 1, 0],
       [0, 1, 0, 1, 0, 1, 0, 1]])
```

[18]:
```
#Create a 3x3 matrix with random integer values. Perform matrix addition,␣
 ↪multilication and subtraction.
o = np.random.randint(1, 10, (3, 3))
p= np.random.randint(1, 10, (3, 3))
```

[21]:
```
np.add(o,p)
```

[21]:
```
array([[11, 13,  8],
       [14, 10, 15],
       [11,  9,  9]])
```

[20]:
```
np.subtract(o,p)
```

[20]:
```
array([[ 3,  1,  4],
       [-2,  4, -3],
       [-3, -1,  3]])
```

[172]:
```
np.dot(o,p)
```

[172]:
```
array([[ 50,  95, 107],
       [ 43,  42,  53],
       [ 55, 157, 172]])
```

[ ]:

[ ]: