Numpy Arrays

In [2]:

```
import numpy as np
```

1-Dimensional Arrays

In [3]:

```
a = [1,2,3,4]
print('a: ',a)



b = np.array(a)
print('b: ',b)
```

```
a:  [1, 2, 3, 4]
b:  [1 2 3 4]
```

In [4]:

```
b
```

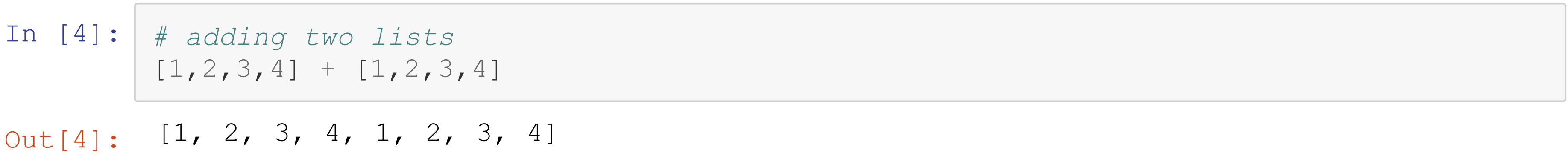
Out[4]:

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

Addition

In [ ]:

```
# adding two lists

[1,2,3,4] + [1,2,3,4]
```

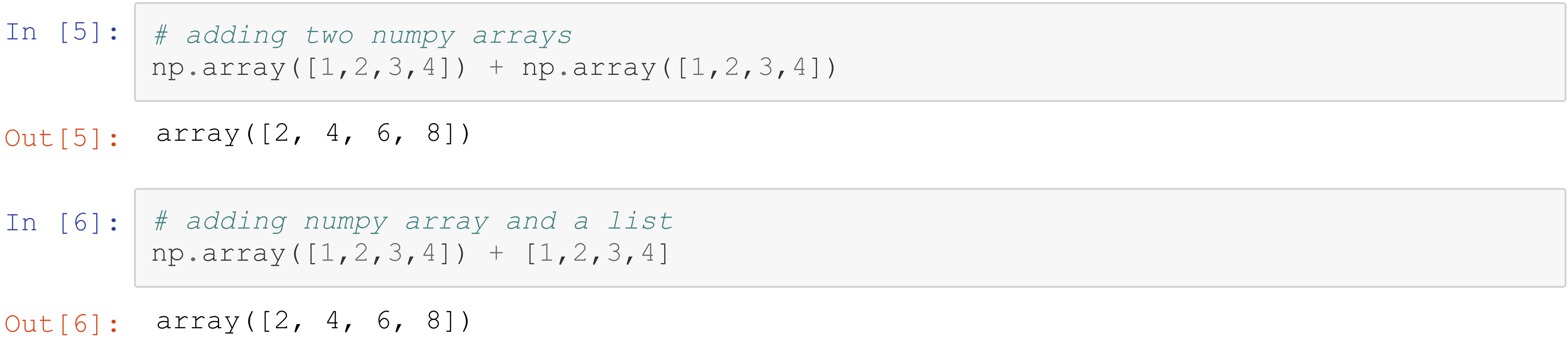


In [4]:

```python
# adding two lists
[1,2,3,4] + [1,2,3,4]
```

Out[4]:

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

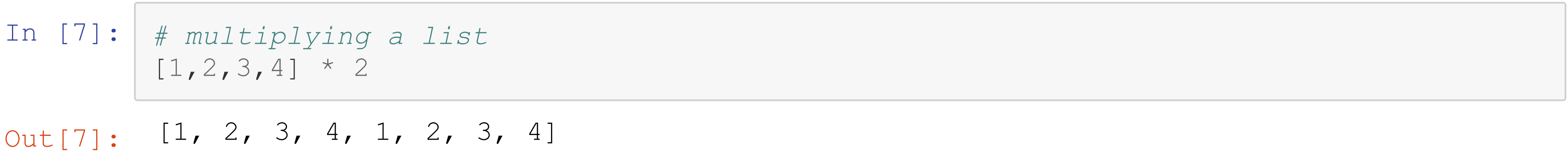


```
In [5]: # adding two numpy arrays
        np.array([1,2,3,4]) + np.array([1,2,3,4])

Out[5]: array([2, 4, 6, 8])

In [6]: # adding numpy array and a list
        np.array([1,2,3,4]) + [1,2,3,4]

Out[6]: array([2, 4, 6, 8])
```

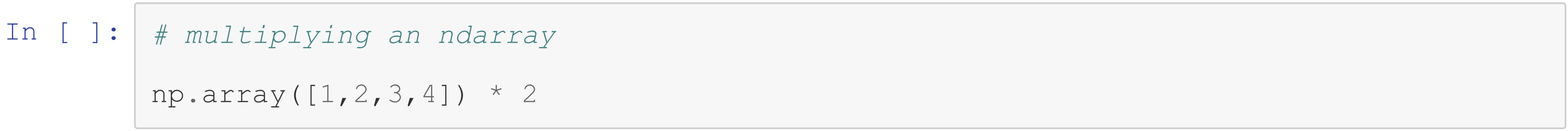
Multiplication

In [ ]:

```
# multiplying a list

[1,2,3,4] * 2
```

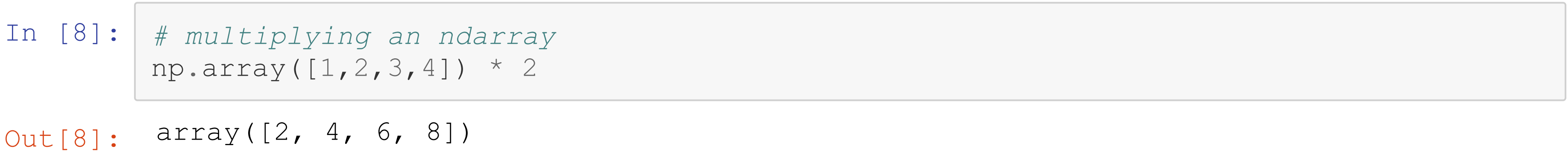


```
In [7]:  # multiplying a list
          [1,2,3,4] * 2

Out[7]:   [1, 2, 3, 4, 1, 2, 3, 4]
```



In [ ]:

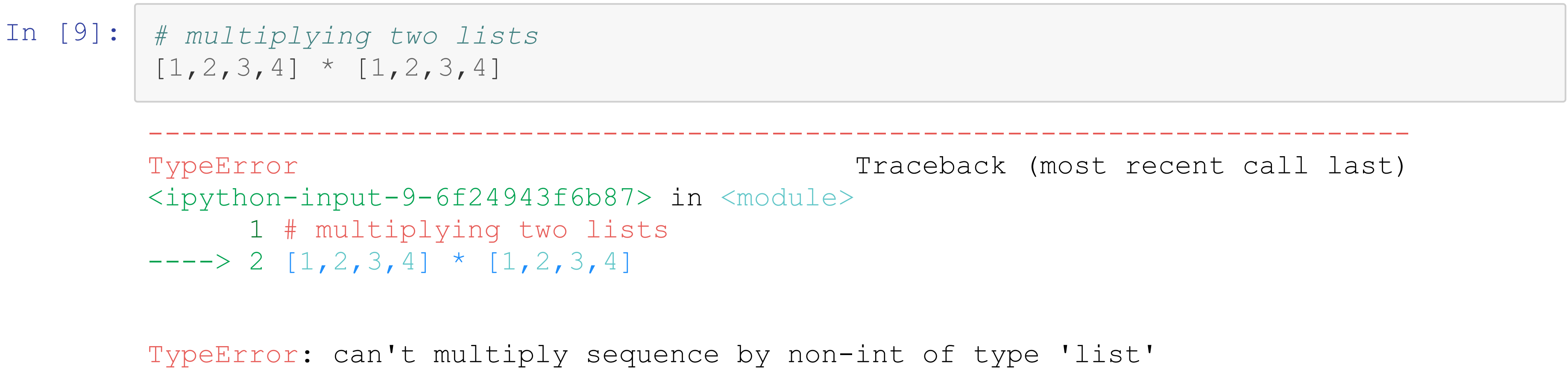
```
# multiplying an ndarray

np.array([1,2,3,4]) * 2
```



```
In [8]:  # multiplying an ndarray
         np.array([1,2,3,4]) * 2

Out[8]:  array([2, 4, 6, 8])
```

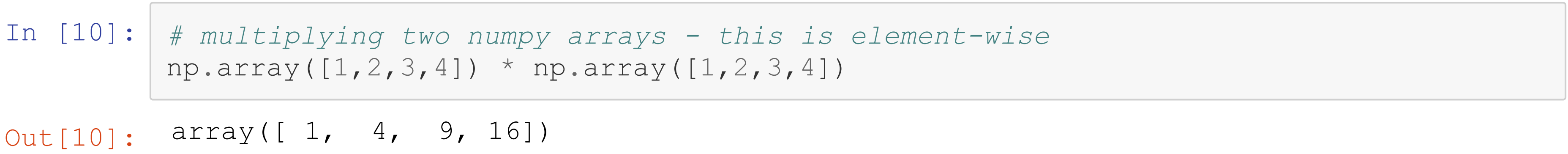
In [ ]:

```
# multiplying two lists

[1,2,3,4] * [1,2,3,4]
```



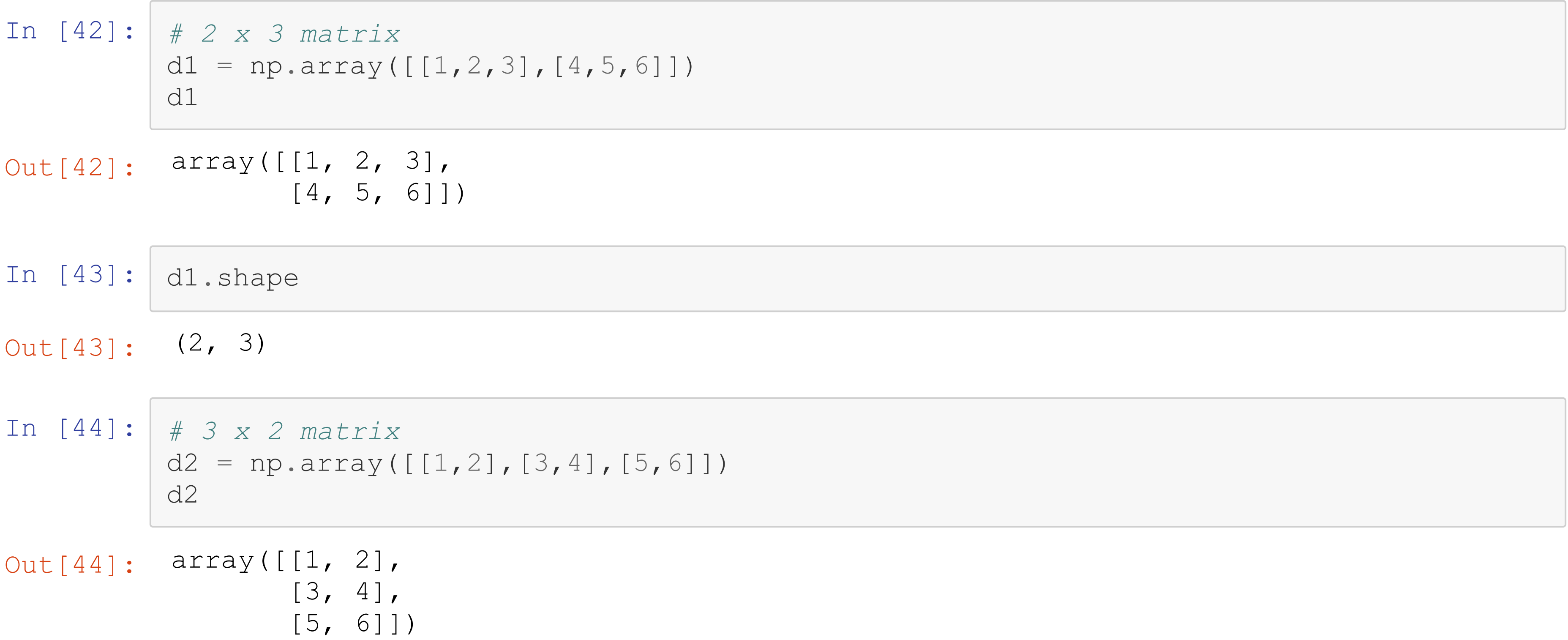
```
In [9]: # multiplying two lists
        [1,2,3,4] * [1,2,3,4]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-9-6f24943f6b87> in <module>
      1 # multiplying two lists
----> 2 [1,2,3,4] * [1,2,3,4]


TypeError: can't multiply sequence by non-int of type 'list'
```



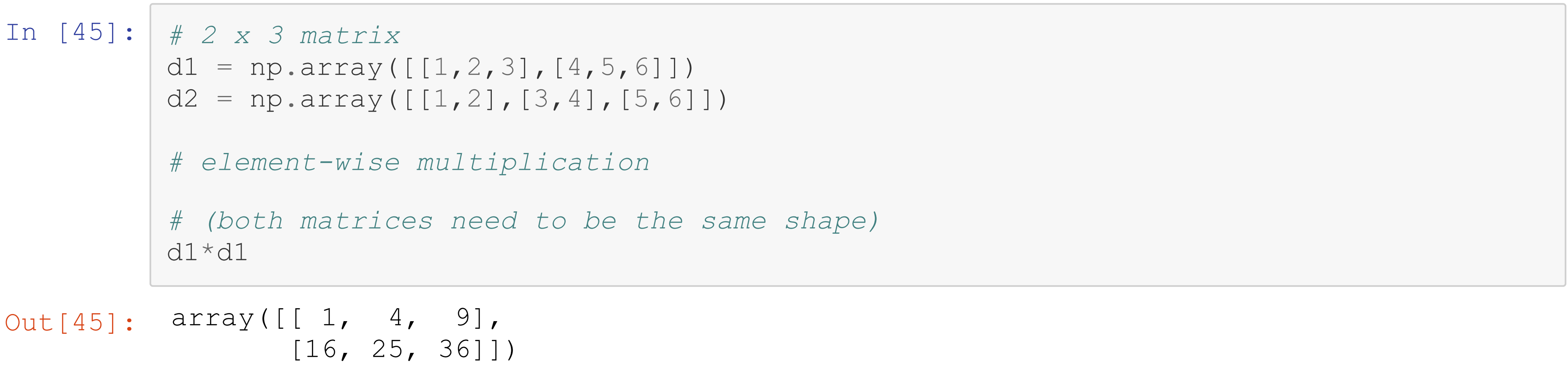
```
In [10]: # multiplying two numpy arrays - this is element-wise
         np.array([1,2,3,4]) * np.array([1,2,3,4])

Out[10]: array([ 1,  4,  9, 16])
```

N-Dimensional Arrays



```
In [42]: # 2 x 3 matrix
         d1 = np.array([[1,2,3],[4,5,6]])
         d1

Out[42]: array([[1, 2, 3],
                [4, 5, 6]])

In [43]: d1.shape

Out[43]: (2, 3)

In [44]: # 3 x 2 matrix
         d2 = np.array([[1,2],[3,4],[5,6]])
         d2

Out[44]: array([[1, 2],
                [3, 4],
                [5, 6]])
```

Element-wise Multiplication

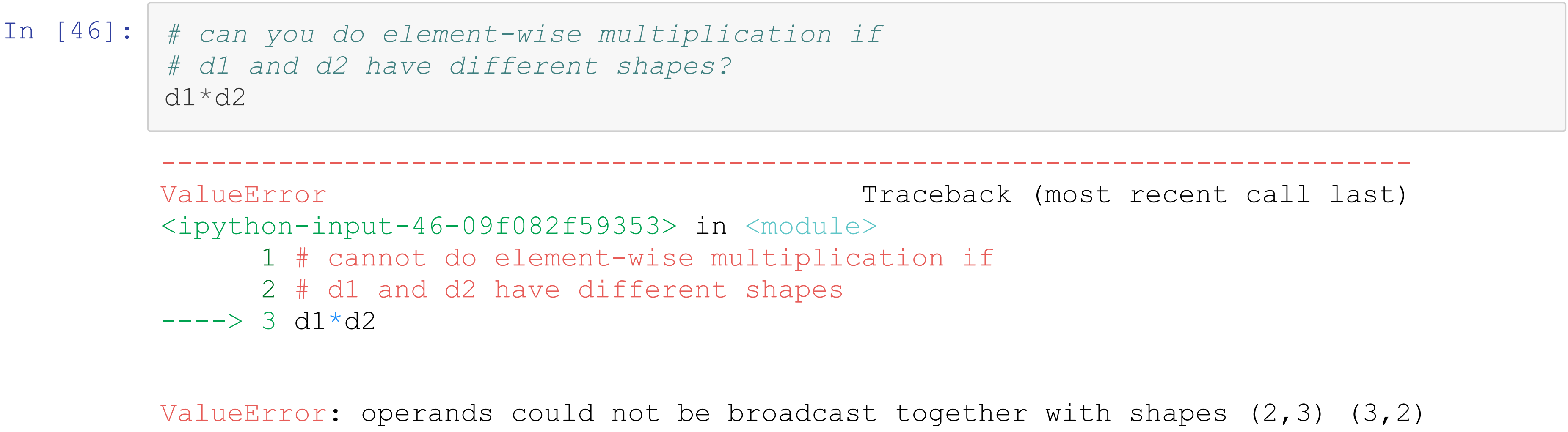


```
# 2 x 3 matrix
d1 = np.array([[1,2,3],[4,5,6]])
d2 = np.array([[1,2],[3,4],[5,6]])

# element-wise multiplication

# (both matrices need to be the same shape)
d1*d1
```

```
array([[ 1,  4,  9],
       [16, 25, 36]])
```

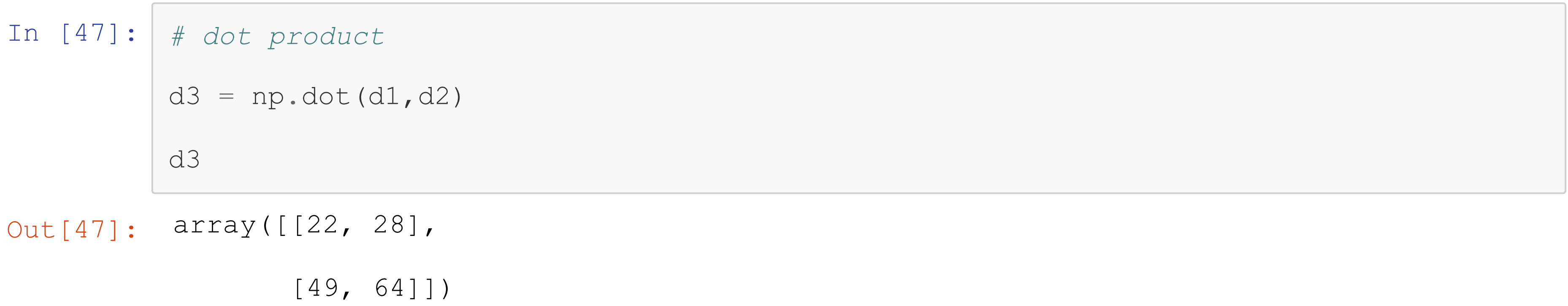
In [ ]:

```
# can you do element-wise multiplication if
# d1 and d2 have different shapes?
d1*d2
```



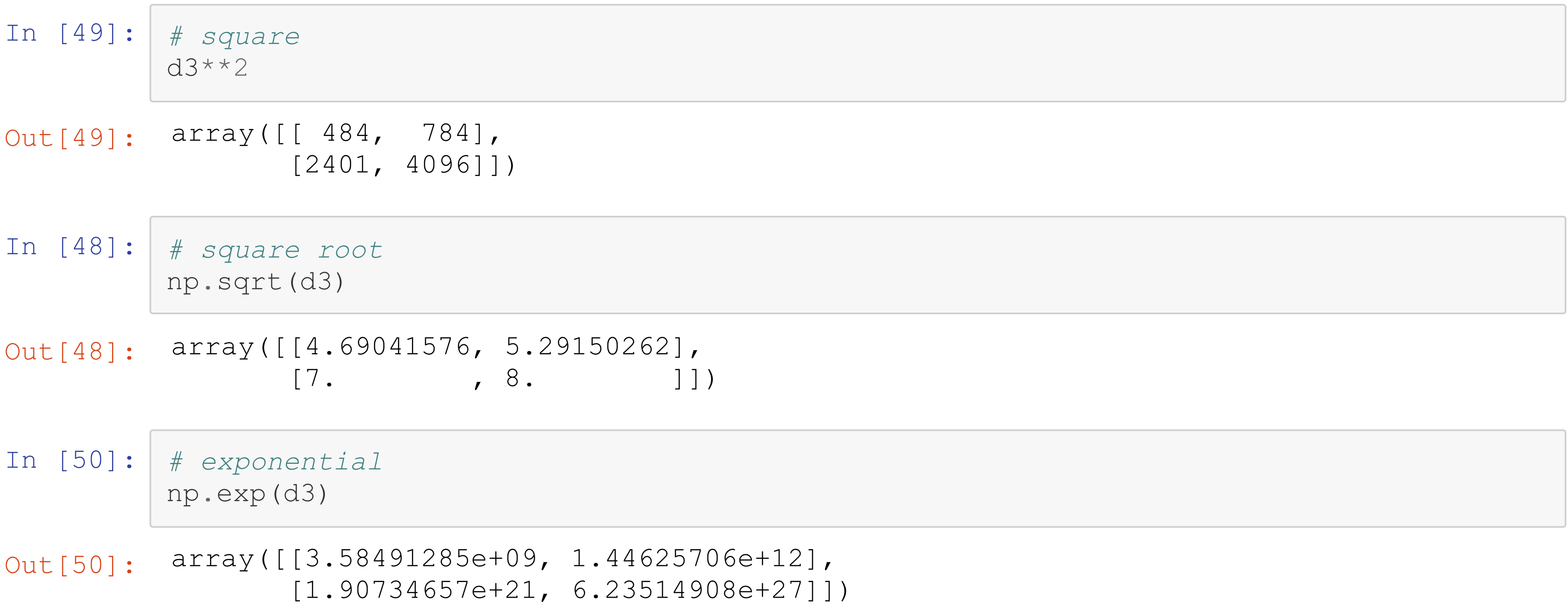
```
In [46]: # can you do element-wise multiplication if
         # d1 and d2 have different shapes?
         d1*d2

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-46-09f082f59353> in <module>
      1 # cannot do element-wise multiplication if
      2 # d1 and d2 have different shapes
----> 3 d1*d2


ValueError: operands could not be broadcast together with shapes (2,3) (3,2)
```

Dot product



```
In [47]: # dot product

         d3 = np.dot(d1,d2)

         d3

Out[47]: array([[22, 28],

                [49, 64]])
```

Mathematical Functions

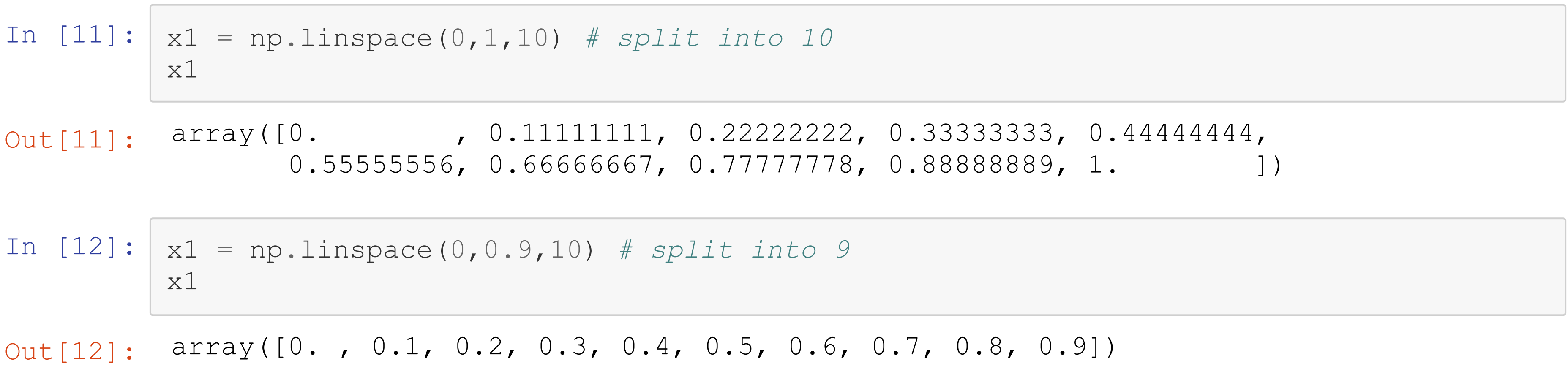


```
In [49]: # square
         d3**2

Out[49]: array([[ 484,  784],
                [2401, 4096]])
```

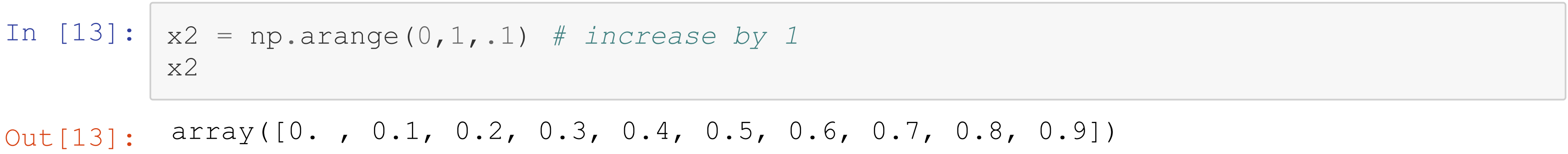
```
In [48]: # square root
         np.sqrt(d3)

Out[48]: array([[4.69041576, 5.29150262],
                [7.        , 8.        ]])
```

```
In [50]: # exponential
         np.exp(d3)

Out[50]: array([[3.58491285e+09, 1.44625706e+12],
                [1.90734657e+21, 6.23514908e+27]])
```

`linspace` and `arange`



```
In [11]: x1 = np.linspace(0,1,10) # split into 10
         x1

Out[11]: array([0.        , 0.11111111, 0.22222222, 0.33333333, 0.44444444,
                0.55555556, 0.66666667, 0.77777778, 0.88888889, 1.        ])

In [12]: x1 = np.linspace(0,0.9,10) # split into 9
         x1

Out[12]: array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9])
```

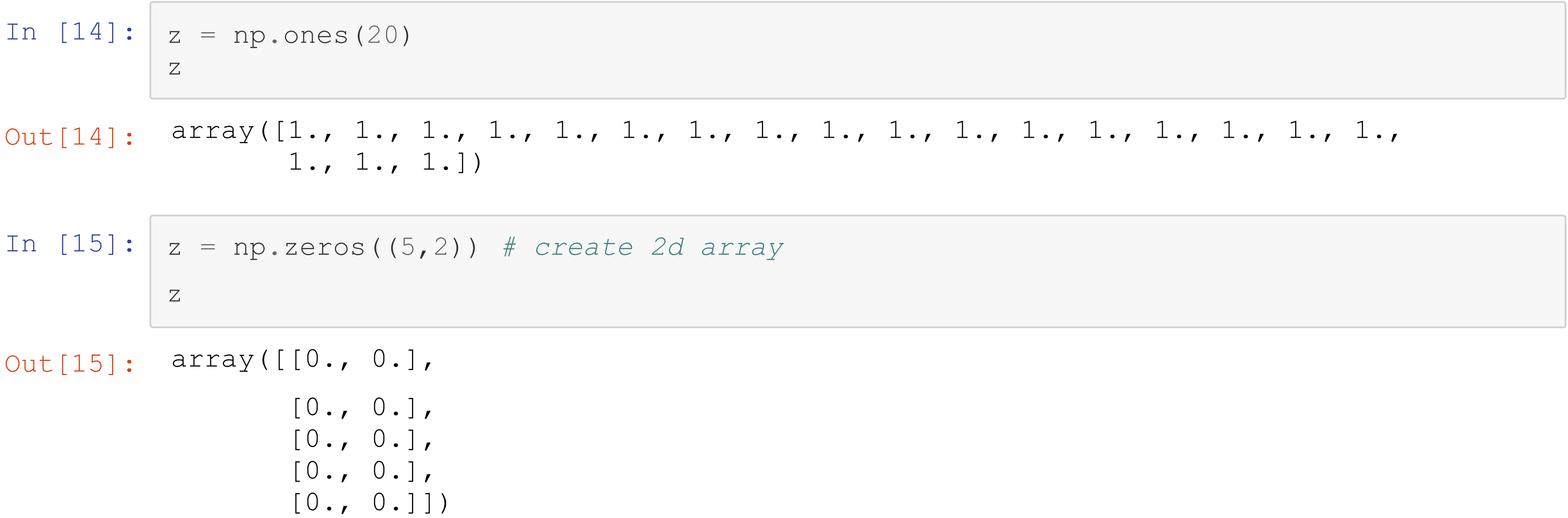


In [13]:

```
x2 = np.arange(0,1,.1) # increase by 1
x2
```

Out[13]:

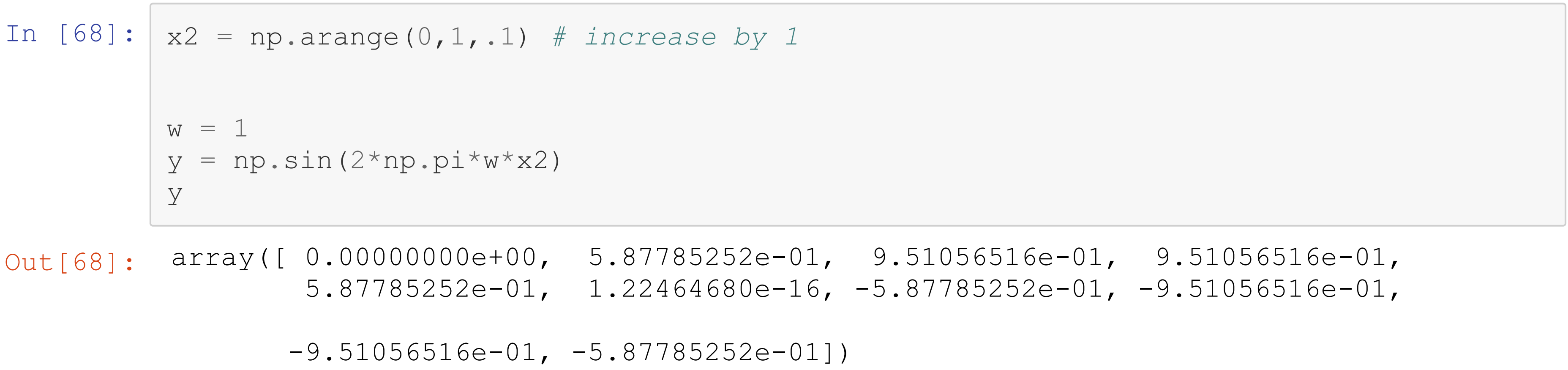
```
array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9])
```



```
In [14]: z = np.ones(20)
         z

Out[14]: array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
                1., 1., 1.])
```

```
In [15]: z = np.zeros((5,2)) # create 2d array
         z

Out[15]: array([[0., 0.],
                [0., 0.],
                [0., 0.],
                [0., 0.],
                [0., 0.]])
```

getting fancy

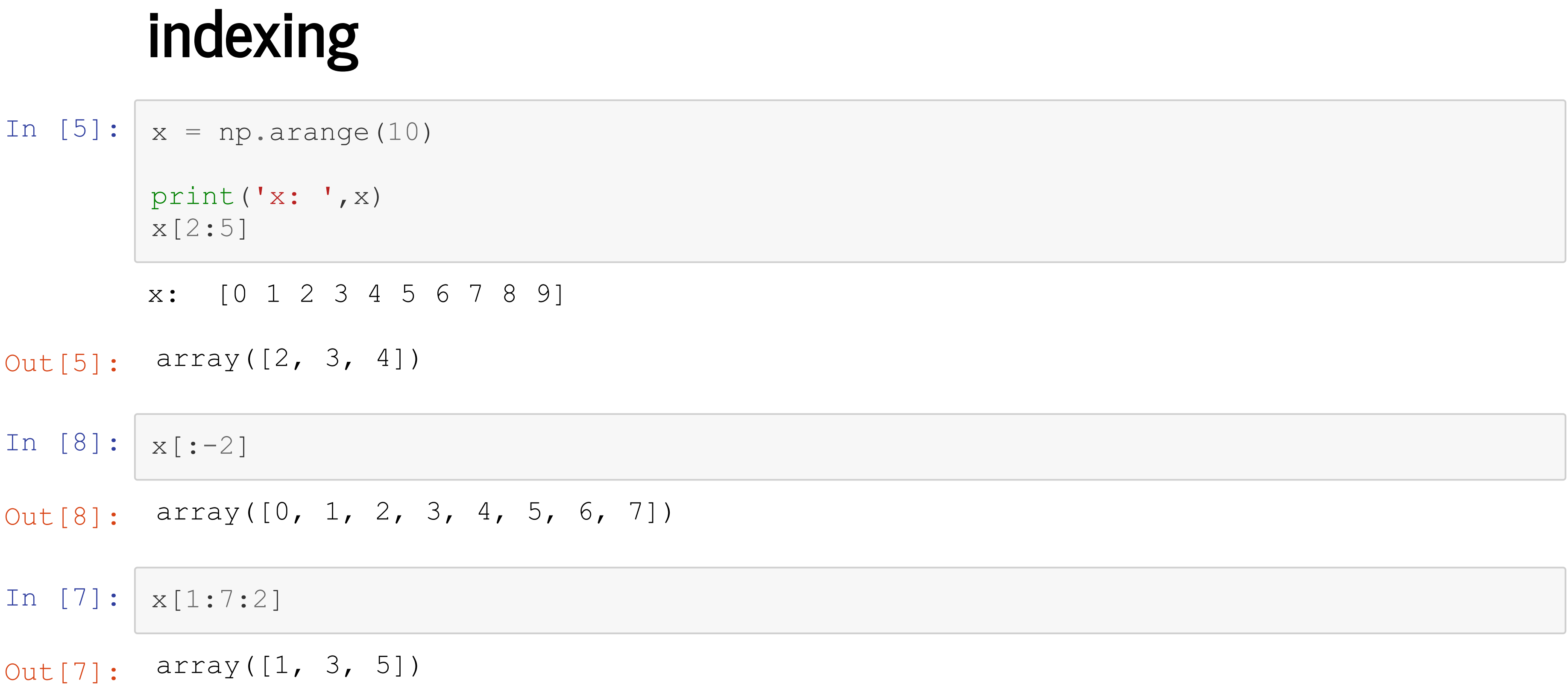


In [68]:

```
x2 = np.arange(0,1,.1) # increase by 1


w = 1
y = np.sin(2*np.pi*w*x2)
y
```

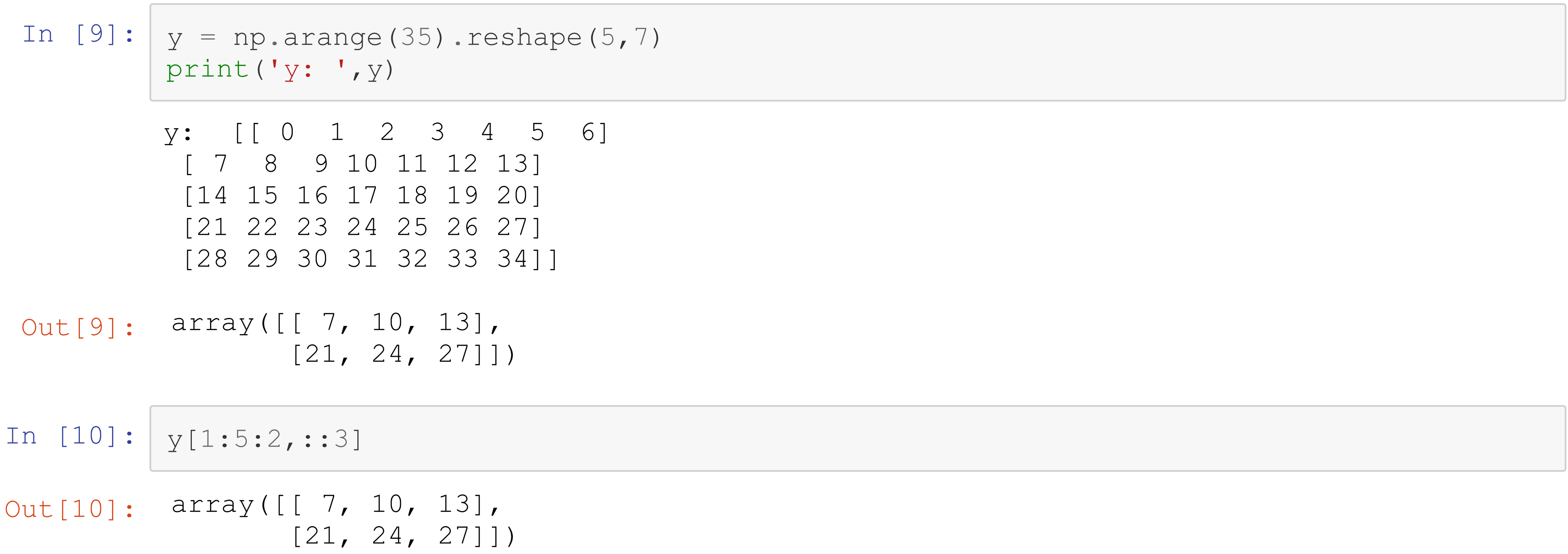
Out[68]:

```
array([ 0.00000000e+00,  5.87785252e-01,  9.51056516e-01,  9.51056516e-01,
        5.87785252e-01,  1.22464680e-16, -5.87785252e-01, -9.51056516e-01,

       -9.51056516e-01, -5.87785252e-01])
```



indexing

```
In [5]:  x = np.arange(10)

         print('x: ',x)
         x[2:5]

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

Out[5]:  array([2, 3, 4])

In [8]:  x[:-2]

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

In [7]:  x[1:7:2]

Out[7]:  array([1, 3, 5])
```



In [9]:

```
y = np.arange(35).reshape(5,7)
print('y: ',y)
```

```
y:  [[ 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 25 26 27]
 [28 29 30 31 32 33 34]]
```

Out[9]:

```
array([[ 7, 10, 13],
       [21, 24, 27]])
```

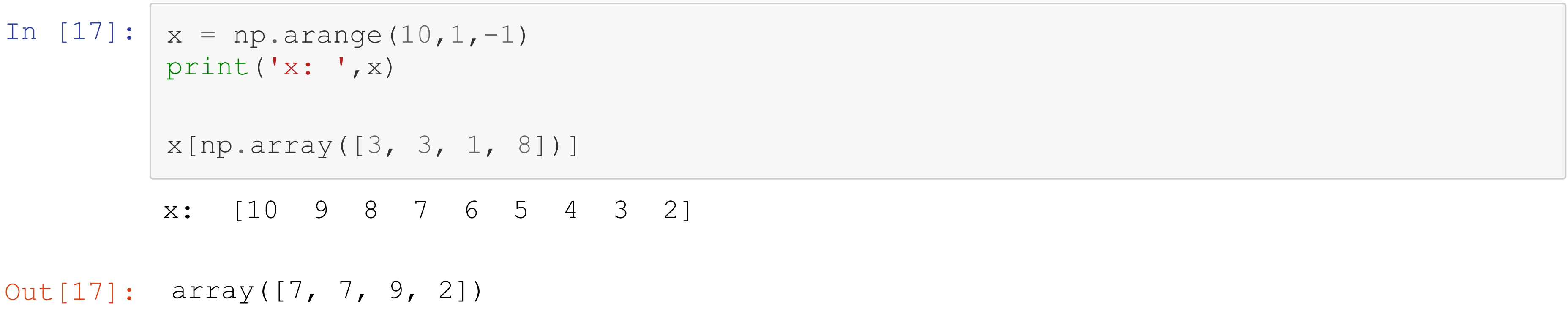
In [10]:

```
y[1:5:2,::3]
```

Out[10]:

```
array([[ 7, 10, 13],
       [21, 24, 27]])
```

Index arrays



In [17]:

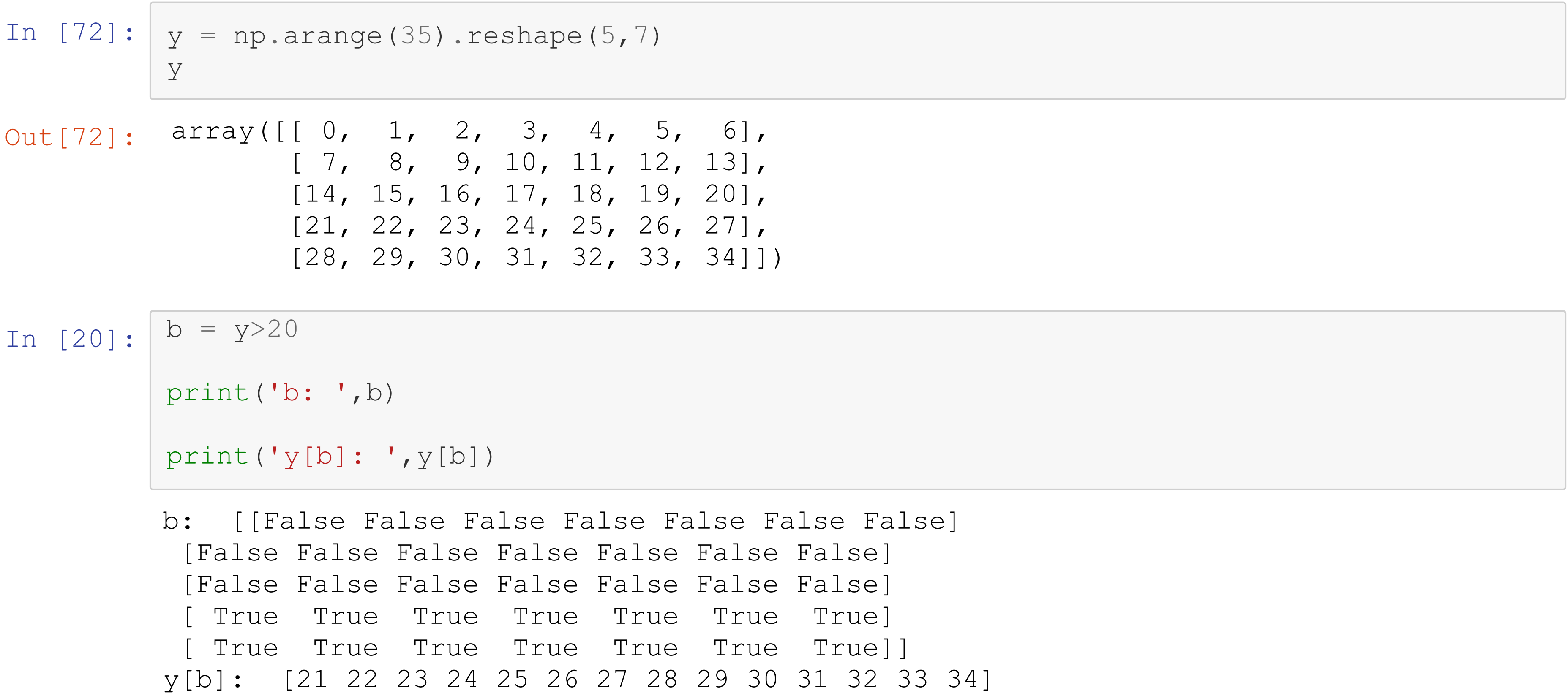
```
x = np.arange(10,1,-1)
print('x: ',x)


x[np.array([3, 3, 1, 8])]
```

```
x:  [10  9  8  7  6  5  4  3  2]
```

Out[17]:

```
array([7, 7, 9, 2])
```

Boolean or “mask” index arrays



```
In [72]: y = np.arange(35).reshape(5,7)
         y

Out[72]: 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, 25, 26, 27],
                [28, 29, 30, 31, 32, 33, 34]])

In [20]: b = y>20

         print('b: ',b)

         print('y[b]: ',y[b])

         b:  [[False False False False False False False]
          [False False False False False False False]
          [False False False False False False False]
          [ True  True  True  True  True  True  True]
          [ True  True  True  True  True  True  True]]
         y[b]:  [21 22 23 24 25 26 27 28 29 30 31 32 33 34]
```

see https://docs.scipy.org/doc/numpy/user/basics.indexing.html for more detail