In [3]:
```
import numpy as np
a = np.array([0, 1, 2, 3])
a
```
Out[3]:
```
array([0, 1, 2, 3])
```
In [4]:
```
L = range(1000)
```
In [5]:
```
%timeit [i**2 for i in L]
```
```
10000 loops, best of 3: 63.6 µs per loop
```
In [6]:
```
a = np.arange(1000)
```
In [7]:
```
%timeit a**2
```
```
100000 loops, best of 3: 1.7 µs per loop
```
In [8]:
```
import numpy as np
```
In [9]:
```
a = np.array([0, 1, 2, 3])
a
```
Out[9]:
```
array([0, 1, 2, 3])
```
In [10]:
```
a.ndim
```
Out[10]:
```
1
```
In [11]:
```
a.shape
```
Out[11]:
```
(4,)
```
In [12]:
```
len(a)
```
Out[12]:
```
4
```
In [13]:
```
b = np.array([[0, 1, 2], [3, 4, 5]])    # 2 x 3 array
b
```
Out[13]:
```
array([[0, 1, 2],
       [3, 4, 5]])
```
In [14]:
```
b.ndim
```
Out[14]:
```
2
```
In [15]:
```
b.shape
```
Out[15]:
```
(2, 3)
```
In [16]:
```
len(b)     # returns the size of the first dimension
```
Out[16]:
```
2
```
In [17]:
```
c = np.array([[[1], [2]], [[3], [4]]])
c
```
Out[17]:
```
array([[[1],
        [2]],

       [[3],
        [4]]])
```

In [18]:
```
c.shape
```

Out[18]:
```
(2, 2, 1)
```

In [19]:
```
a = np.arange(10) # 0 .. n-1  (!)
a
```

Out[19]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [20]:
```
b = np.arange(1, 9, 2) # start, end (exclusive), step
b
```

Out[20]:
```
array([1, 3, 5, 7])
```

In [21]:
```
c = np.linspace(0, 1, 6)   # start, end, num-points
c
```

Out[21]:
```
array([ 0. ,  0.2,  0.4,  0.6,  0.8,  1. ])
```

In [22]:
```
d = np.linspace(0, 1, 5, endpoint=False)
d
```

Out[22]:
```
array([ 0. ,  0.2,  0.4,  0.6,  0.8])
```

In [23]:
```
a = np.ones((3, 3))  # reminder: (3, 3) is a tuple
a
```

Out[23]:
```
array([[ 1.,  1.,  1.],
       [ 1.,  1.,  1.],
       [ 1.,  1.,  1.]])
```

In [24]:
```
b = np.zeros((2, 2))
b
```

Out[24]:
```
array([[ 0.,  0.],
       [ 0.,  0.]])
```

In [25]:
```
c = np.eye(3)
c
```

Out[25]:
```
array([[ 1.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  1.]])
```

In [26]:
```
d = np.diag(np.array([1, 2, 3, 4]))
d
```

Out[26]:
```
array([[1, 0, 0, 0],
       [0, 2, 0, 0],
       [0, 0, 3, 0],
       [0, 0, 0, 4]])
```

In [27]:
```
a = np.random.rand(4)       # uniform in [0, 1]
a
```

Out[27]:
```
array([ 0.2300723 ,  0.73602459,  0.32483886,  0.12833181])
```

In [28]:
```
b = np.random.randn(4)      # Gaussian
b
```

Out[28]:
```
array([-0.10174901, -0.14529344,  0.32421093,  1.60137082])
```

In [29]:
```
np.random.seed(1234)        # Setting the random seed
```

In [30]:
```
a = np.array([1, 2, 3])
```

```
a.dtype
```

Out[30]:
```
dtype('int64')
```

In [31]:
```
b = np.array([1., 2., 3.])
b.dtype
```

Out[31]:
```
dtype('float64')
```

In [32]:
```
c = np.array([1, 2, 3], dtype=float)
c.dtype
```

Out[32]:
```
dtype('float64')
```

In [33]:
```
a = np.ones((3, 3))
a.dtype
```

Out[33]:
```
dtype('float64')
```

In [34]:
```
d = np.array([1+2j, 3+4j, 5+6*1j])
d.dtype
```

Out[34]:
```
dtype('complex128')
```

In [35]:
```
e = np.array([True, False, False, True])
e.dtype
```

Out[35]:
```
dtype('bool')
```

In [36]:
```
f = np.array(['Bonjour', 'Hello', 'Hallo',])
f.dtype      # <--- strings containing max. 7 letters
```

Out[36]:
```
dtype('S7')
```

In [38]:
```
a = np.arange(10)
a
```

Out[38]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [39]:
```
a[0], a[2], a[-1]
```

Out[39]:
```
(0, 2, 9)
```

In [40]:
```
a[::-1]
```

Out[40]:
```
array([9, 8, 7, 6, 5, 4, 3, 2, 1, 0])
```

In [41]:
```
a = np.diag(np.arange(3))
a
```

Out[41]:
```
array([[0, 0, 0],
       [0, 1, 0],
       [0, 0, 2]])
```

In [42]:
```
a[1, 1]
```

Out[42]:
```
1
```

In [43]:
```
a[2, 1] = 10 # third line, second column
a
```

Out[43]:
```
array([[ 0,  0,  0],
       [ 0,  1,  0],
       [ 0, 10,  2]])
```

In [44]:
```
a[1]
```

Out[44]:
```
array([0, 1, 0])
```

In [45]:
```
a = np.arange(10)
a
```

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

In [46]:
```
a[2:9:3] # [start:end:step]
```

Out[46]:
```
array([2, 5, 8])
```

In [47]:
```
a[:4]
```

Out[47]:
```
array([0, 1, 2, 3])
```

In [48]:
```
a[1:3]
```

Out[48]:
```
array([1, 2])
```

In [49]:
```
a[::2]
```

Out[49]:
```
array([0, 2, 4, 6, 8])
```

In [50]:
```
a[3:]
```

Out[50]:
```
array([3, 4, 5, 6, 7, 8, 9])
```

In [51]:
```
from IPython.display import Image
Image(filename='images/numpy_indexing.png')
```

```
---------------------------------------------------------------------------
IOError                                   Traceback (most recent call last)
<ipython-input-51-ef00be976d20> in <module>()
      1 from IPython.display import Image
----> 2 Image(filename='images/numpy_indexing.png')

/usr/lib/python2.7/dist-packages/IPython/core/display.pyc in __init__(self, data, url, filename, format, embed, width, height, retina)
    599         self.height = height
    600         self.retina = retina
--> 601         super(Image, self).__init__(data=data, url=url, filename=filename)
    602 
    603         if retina:

/usr/lib/python2.7/dist-packages/IPython/core/display.pyc in __init__(self, data, url, filename)
    303         self.filename = None if filename is None else unicode(filename)
    304 
--> 305         self.reload()
    306 
    307     def reload(self):

/usr/lib/python2.7/dist-packages/IPython/core/display.pyc in reload(self)
    621         """Reload the raw data from file or URL."""
    622         if self.embed:
--> 623             super(Image,self).reload()
    624             if self.retina:
    625                 self._retina_shape()

/usr/lib/python2.7/dist-packages/IPython/core/display.pyc in reload(self)
    308         """Reload the raw data from file or URL."""
    309         if self.filename is not None:
--> 310             with open(self.filename, self._read_flags) as f:
    311                 self.data = f.read()
    312         elif self.url is not None:

IOError: [Errno 2] No such file or directory: u'images/numpy_indexing.png'
```

```
IOError: [Errno 2] No such file or directory: u'images/numpy_indexing.png'
```

In []:

```
```