# Creating NumPy Arrays </h2 >

The following ways are commonly used when you know the size of the array beforehand:

- `np.ones()` : Create array of 1s
- `np.zeros()` : Create array of 0s
- `np.random.random()` : Create array of random numbers
- `np.arange()` : Create array with increments of a fixed step size
- `np.linspace()` : Create array of fixed length

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

Tip: Use help to see the syntax when required

In [2]:
```
help(np.ones)
```

```
Help on function ones in module numpy:

ones(shape, dtype=None, order='C')
    Return a new array of given shape and type, filled with ones.

    Parameters
    ----------
    shape : int or sequence of ints
        Shape of the new array, e.g., ``(2, 3)`` or ``2``.
    dtype : data-type, optional
        The desired data-type for the array, e.g., `numpy.int8`.  Default is
        `numpy.float64`.
    order : {'C', 'F'}, optional, default: C
        Whether to store multi-dimensional data in row-major
        (C-style) or column-major (Fortran-style) order in
        memory.

    Returns
    -------
    out : ndarray
        Array of ones with the given shape, dtype, and order.
```

```
See Also
--------
ones_like : Return an array of ones with shape and type of input.
empty : Return a new uninitialized array.
zeros : Return a new array setting values to zero.
full : Return a new array of given shape filled with value.


Examples
--------
>>> np.ones(5)
array([1., 1., 1., 1., 1.])

>>> np.ones((5,), dtype=int)
array([1, 1, 1, 1, 1])

>>> np.ones((2, 1))
array([[1.],
       [1.]])

>>> s = (2,2)
>>> np.ones(s)
array([[1.,  1.],
       [1.,  1.]])
```

Creating a 1 D array of ones

In [3]:
```
arr = np.ones(5)
arr
```

Out[3]:
```
array([1., 1., 1., 1., 1.])
```

Notice that, by default, numpy creates data type = float64

In [4]:
```
arr.dtype
```

Out[4]:
```
dtype('float64')
```

Can provide dtype explicitly using dtype

In [5]:
```
arr = np.ones(5, dtype=int)
```

```
In [5]: arr
```

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

```
In [6]: arr.dtype
```

```
Out[6]: dtype('int64')
```

Creating a 5 x 3 array of ones

```
In [7]: np.ones((5,3))
```

```
Out[7]: array([[1., 1., 1.],
               [1., 1., 1.],
               [1., 1., 1.],
               [1., 1., 1.],
               [1., 1., 1.]])
```

Creating array of zeros

```
In [8]: np.zeros(5)
```

```
Out[8]: array([0., 0., 0., 0., 0.])
```

```
In [9]: # convert the type into integer.
        np.zeros(5, dtype=int)
```

```
Out[9]: array([0, 0, 0, 0, 0])
```

```
In [12]: # Create a list of integers range between 1 to 5.
         list(range(1,5))
```

```
Out[12]: [1, 2, 3, 4]
```

```
In [13]: np.arange(3)
```

```
Out[13]: array([0, 1, 2])
```

```
In [14]: np.arange(3.0)
```

Out[14]:
```
array([0., 1., 2.])
```

Notice that 3 is included, 35 is not, as in standard python lists

From 3 to 35 with a step of 2

In [20]:
```
np.arange(3,35,2)
```

Out[20]:
```
array([ 3,  5,  7,  9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33])
```

Array of random numbers

In [21]:
```
np.random.randint(2, size=10)
```

Out[21]:
```
array([0, 1, 0, 1, 1, 1, 0, 0, 0, 0])
```

In [24]:
```
np.random.randint(3,5, size=10)
```

Out[24]:
```
array([3, 3, 3, 3, 4, 3, 3, 3, 3, 4])
```

2D Array of random numbers

In [25]:
```
np.random.random([3,4])
```

Out[25]:
```
array([[0.37947795, 0.50446351, 0.76204337, 0.23268129],
       [0.49530063, 0.37298231, 0.17830691, 0.9400508 ],
       [0.18746889, 0.99395211, 0.03729134, 0.16021317]])
```

Sometimes, you know the length of the array, not the step size

Array of length 20 between 1 and 10

In [27]:
```
np.linspace(1,10,20)
```

Out[27]:
```
array([ 1.        ,  1.47368421,  1.94736842,  2.42105263,  2.89473684,
        3.36842105,  3.84210526,  4.31578947,  4.78947368,  5.26315789,
        5.73684211,  6.21052632,  6.68421053,  7.15789474,  7.63157895,
        8.10526316,  8.57894737,  9.05263158,  9.52631579, 10.        ])
```

# Exercises

Apart from the methods mentioned above, there are a few more NumPy functions that you can use to create special NumPy arrays:

- `np.full()` : Create a constant array of any number ‘n’
- `np.tile()` : Create a new array by repeating an existing array for a particular number of times
- `np.eye()` : Create an identity matrix of any dimension
- `np.random.randint()` : Create a random array of integers within a particular range

In [ ]: