# NumPy Datatypes

NumPy boasts a broad range of numerical datatypes in comparison with vanilla Python. This extended datatype support is useful for dealing with different kinds of signed and unsigned integer and floating-point numbers as well as booleans and complex numbers for scientific computation. NumPy datatypes include the **bool_**, **int**(8,16,32,64), **uint**(8,16,32,64), **float**(16,32,64), **complex**(64,128) as well as the **int_**, **float_**, and **complex_**, to mention just a few.

The datatypes with a _ appended are base Python datatypes converted to NumPy datatypes. The parameter **dtype** is used to assign a datatype to a NumPy function. The default NumPy type is **float_**. Also, NumPy infers contiguous arrays of the same type.

Let's explore a bit with NumPy datatypes:

```
# ints
my_ints = np.array([3, 7, 9, 11])
my_ints.dtype
'Output': dtype('int64')

# floats
my_floats = np.array([3., 7., 9., 11.])
my_floats.dtype
'Output': dtype('float64')

# non-contiguous types - default: float
my_array = np.array([3., 7., 9, 11])
my_array.dtype
'Output': dtype('float64')

# manually assigning datatypes
my_array = np.array([3, 7, 9, 11], dtype="float64")
my_array.dtype
'Output': dtype('float64')
```

# Indexing + Fancy Indexing (1-D)

We can index a single element of a NumPy 1-D array similar to how we index a Python list.

```
# create a random numpy 1-D array
my_array = np.random.rand(10)
my_array
'Output': array([ 0.7736445 ,  0.28671796,  0.61980802,  0.42110553,
                  0.86091567,  0.93953255,  0.300224  ,  0.56579416,
                  0.58890282,   0.97219289])
# index the first element
my_array[0]
'Output': 0.77364449999999996
# index the last element
my_array[-1]
'Output': 0.97219288999999998
```

Fancy indexing in NumPy is an advanced mechanism for indexing array elements based on integers or boolean. This technique is also called *masking*.

# Boolean Mask

Let's index all the even integers in the array using a boolean mask.

```
# create 10 random integers between 1 and 20
my_array = np.random.randint(1, 20, 10)
my_array
'Output': array([14,  9,  3, 19, 16,  1, 16,  5, 13,  3])
# index all even integers in the array using a boolean mask
my_array[my_array % 2 == 0]
'Output': array([14, 16, 16])
```

Observe that the code my_array % 2 == 0 outputs an array of booleans.

```
my_array % 2 == 0
'Output': array([ True, False, False, False,  True, False,  True, False,
False, False], dtype=bool)
```