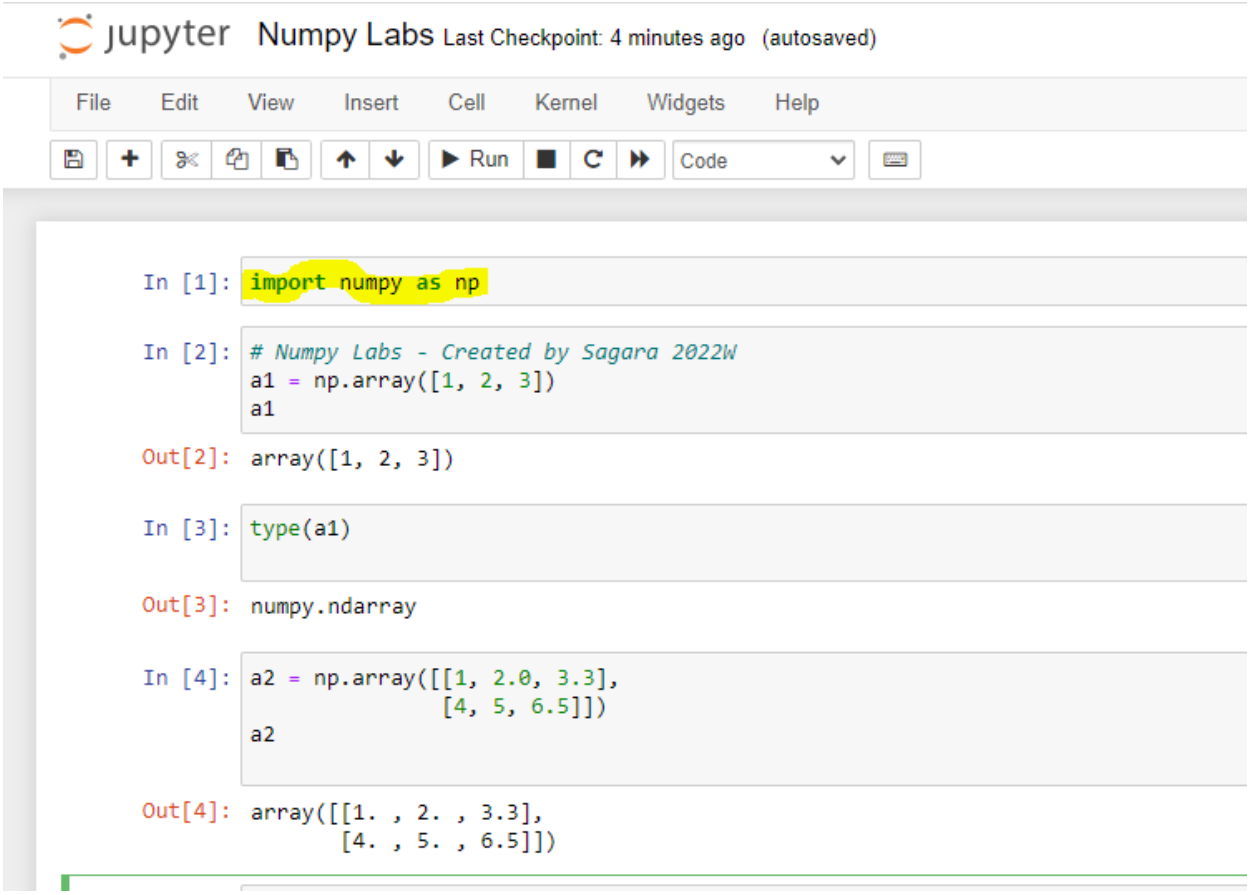
Week 6 - LAB

Python Libraries – Numpy Library



```
In [5]: a3 = np.array([[[1, 2, 3],
                        [4, 5, 6],
                        [7, 8, 9]],
                       [[10, 11, 12],
                        [13, 14, 15],
                        [16, 17, 18]]])
        a3

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

               [[10, 11, 12],
                [13, 14, 15],
                [16, 17, 18]]])

In [6]: a1.shape

Out[6]: (3,)
```

```
In [6]: a1.shape

Out[6]: (3,)

In [7]: a2.shape

Out[7]: (2, 3)

In [8]: a1.ndim, a2.ndim, a3.ndim

Out[8]: (1, 2, 3)

In [9]: a1.dtype, a2.dtype, a3.dtype

Out[9]: (dtype('int32'), dtype('float64'), dtype('int32'))

In [10]: a1.size, a2.size, a3.size

Out[10]: (3, 6, 18)
```

2. Creating arrays

In [12]:
```
# Create a DataFrame from a NumPy array
import pandas as pd

df = pd.DataFrame(a2)
df
```

Out[12]:

	0	1	2
0	1.0	2.0	3.3
1	4.0	5.0	6.5

In [13]:
```
sample_array = np.array([1, 2, 3])
sample_array
```

Out[13]:
```
array([1, 2, 3])
```

In [14]:
```
ones = np.ones((2, 3))
```

In [15]:
```
ones
```

Out[15]:
```
array([[1., 1., 1.],
       [1., 1., 1.]])
```

In [16]:
```
zeros = np.zeros((2, 3))
zeros
```

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

In [17]:
```
range_array = np.arange(0, 10, 2)
range_array
```

Out[17]:
```
array([0, 2, 4, 6, 8])
```

In [18]:
```
random_array = np.random.randint(0, 10, size=(3, 5))
random_array
```

Out[18]:
```
array([[9, 1, 3, 3, 8],
       [0, 8, 8, 7, 8],
       [7, 9, 6, 9, 4]])
```

Manipulating & comparing arrays

```
In [19]: a1
Out[19]: array([1, 2, 3])

In [20]: ones = np.ones(3)
         ones
Out[20]: array([1., 1., 1.])

In [21]: a1 + ones
Out[21]: array([2., 3., 4.])

In [22]: a1 - ones
Out[22]: array([0., 1., 2.])

In [23]: a1 * ones
Out[23]: array([1., 2., 3.])
```

```
In [24]: a1
Out[24]: array([1, 2, 3])

In [25]: a2
Out[25]: array([[1. , 2. , 3.3],
                [4. , 5. , 6.5]])

In [26]: a1 * a2
Out[26]: array([[ 1. ,  4. ,  9.9],
                [ 4. , 10. , 19.5]])
```

```
In [27]: a2 / a1

Out[27]: array([[1.        , 1.        , 1.1       ],
                [4.        , 2.5       , 2.16666667]])

In [28]: a2//a1

Out[28]: array([[1., 1., 1.],
                [4., 2., 2.]])
```

Aggregation

Aggregation = performing the same operation on a number of things

```
In [29]: listy_list = [1, 2, 3]
         type(listy_list)

Out[29]: list

In [30]: sum(listy_list)

Out[30]: 6

In [31]: sum(a1)

Out[31]: 6

In [32]: np.sum(a1)

Out[32]: 6
```

Use Python's methods (sum()) on Python datatypes and use NumPy's methods on NumPy arrays (np.sum()).

```
In [33]: # Creative a massive NumPy array
         massive_array = np.random.random(100000)
         massive_array.size

Out[33]: 100000

In [34]: massive_array[:10]

Out[34]: array([0.7138599 , 0.47519821, 0.33322304, 0.61387367, 0.55523248,
                0.44881566, 0.34278179, 0.62358912, 0.75760114, 0.40712912])

In [35]: %timeit sum(massive_array) # Python's sum()
         %timeit np.sum(massive_array) # NumPy's np.sum()

         16 ms ± 403 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
         87.5 µs ± 21.8 µs per loop (mean ± std. dev. of 7 runs, 10000 loops each)
```

Standard deviation and **variance** are measures of 'spread' of data.

The higher standard deviation and the higher variance of data, the more spread out the values are.

The lower standard deviation and lower variance, the less spread out the values are.

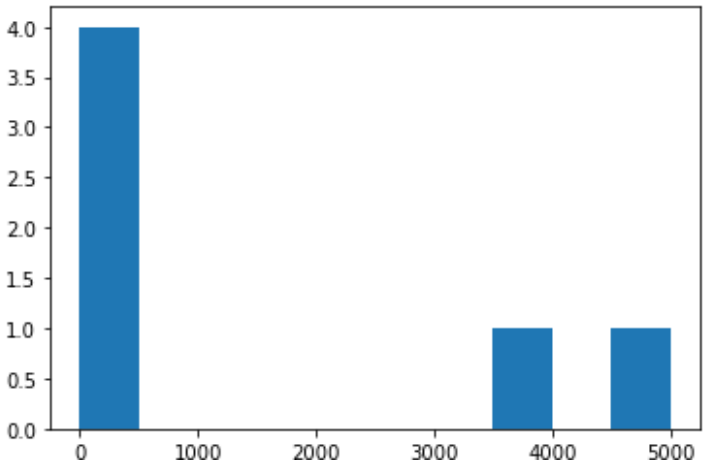
```
In [36]: a2

Out[36]: array([[1. , 2. , 3.3],
                [4. , 5. , 6.5]])

In [37]: np.mean(a2)

Out[37]: 3.6333333333333333

In [38]: np.max(a2)

Out[38]: 6.5

In [39]: np.min(a2)

Out[39]: 1.0

In [40]: np.std(a2)

Out[40]: 1.8226964152656422

In [41]: np.var(a2)

Out[41]: 3.3222222222222224
```

High Variance & Low Variance

```
In [42]: # Demo of std and var
         high_var_array = np.array([1, 100, 200, 300, 4000, 5000])
         low_var_array = np.array([2, 4, 6, 8, 10])

In [43]: np.var(high_var_array), np.var(low_var_array)

Out[43]: (4296133.472222221, 8.0)

In [44]: np.std(high_var_array), np.std(low_var_array)

Out[44]: (2072.711623024829, 2.8284271247461903)

In [45]: np.mean(high_var_array), np.mean(low_var_array)

Out[45]: (1600.1666666666667, 6.0)
```

In [46]:
```
%matplotlib inline
import matplotlib.pyplot as plt
plt.hist(high_var_array)
plt.show()
```

```
Matplotlib is building the font cache; this may take a moment.
```



In [47]:
```
plt.hist(low_var_array)
plt.show()
```