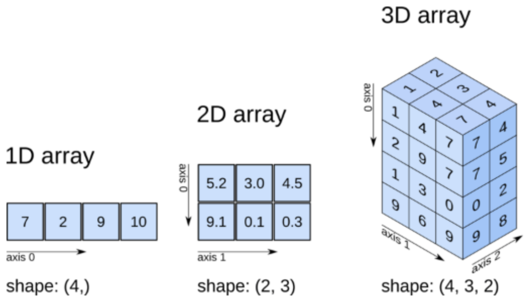
Knowledge Organiser: NumPy

NumPy stands for Numerical Python. It is a Python library that performs numerical calculations.

NumPy is built on linear algebra. It's about matrices and vectors and performing the mathematical calculations on them.

The key concept in NumPy is the *NumPy array* data type. A NumPy array may have one or more dimensions:

- One dimension arrays (1D) represent vectors.
- Two-dimensional arrays (2D) represent matrices.
- And higher dimensional arrays represent tensors.



SCALAR MATH

`np.add(arr,1)` - Add **1** to each array element

`np.subtract(arr,2)` - Subtract **2** from each array element

`np.multiply(arr,3)` - Multiply each array element by **3**

`np.divide(arr,4)` - Divide each array element by **4** (returns **np.nan** for division by zero)

`np.power(arr,5)` - Raise each array element to the **5**th power

CREATING ARRAYS

`np.array([1,2,3])` - One dimensional array

`np.array([(1,2,3),(4,5,6)])` - Two dimensional array

`np.zeros(3)` - 1D array of length **3** all values **0**

`np.ones((3,4))` - **3x4** array with all values **1**

`np.eye(5)` - **5x5** array of **0** with **1** on diagonal (Identity matrix)

`np.linspace(0,100,6)` - Array of **6** evenly divided values from **0** to **100**

`np.arange(0,10,3)` - Array of values from **0** to less than **10** with step **3** (eg **[0,3,6,9]**)

`np.full((2,3),8)` - **2x3** array with all values **8**

`np.random.rand(4,5)` - **4x5** array of random floats between **0-1**

`np.random.rand(6,7)*100` - **6x7** array of random floats between **0-100**

`np.random.randint(5,size=(2,3))` - **2x3** array with random ints between **0-4**

INSPECTING PROPERTIES

`arr.size` - Returns number of elements in **arr**

`arr.shape` - Returns dimensions of **arr** (rows, columns)

`arr.dtype` - Returns type of elements in **arr**

`arr.astype(dtype)` - Convert **arr** elements to type **dtype**

`arr.tolist()` - Convert **arr** to a Python list

`np.info(np.eye)` - View documentation for **np.eye**

COPYING/SORTING/RESHAPING

`np.copy(arr)` - Copies **arr** to new memory

`arr.view(dtype)` - Creates view of **arr** elements with type **dtype**

`arr.sort()` - Sorts **arr**

`arr.sort(axis=0)` - Sorts specific axis of **arr**

`two_d_arr.flatten()` - Flattens 2D array **two_d_arr** to 1D

`arr.T` - Transposes **arr** (rows become columns and vice versa)

`arr.reshape(3,4)` - Reshapes **arr** to **3** rows, **4** columns without changing data

`arr.resize((5,6))` - Changes **arr** shape to **5x6** and fills new values with **0**

INDEXING/SLICING/SUBSETTING

`arr[5]` - Returns the element at index **5**

`arr[2,5]` - Returns the 2D array element on index **[2][5]**

`arr[1]=4` - Assigns array element on index **1** the value **4**

`arr[1,3]=10` - Assigns array element on index **[1][3]** the value **10**

`arr[0:3]` - Returns the elements at indices **0,1,2** (On a 2D array: returns rows **0,1,2**)

`arr[0:3,4]` - Returns the elements on rows **0,1,2** at column **4**

`arr[:2]` - Returns the elements at indices **0,1** (On a 2D array: returns rows **0,1**)