In [1]:
```
# Import NumPy as its abbreviation 'np'
```

In [2]:
```
# Create a 1-dimensional NumPy array using np.array()


# Create a 2-dimensional NumPy array using np.array()


# Create a 3-dimensional Numpy array using np.array()
```

Now we've you've created 3 different arrays, let's find details about them.

Find the shape, number of dimensions, data type, size and type of each array.

In [3]:
```
# Attributes of 1-dimensional array (shape,
# number of dimensions, data type, size and type)
```

In [4]:
```
# Attributes of 2-dimensional array
```

In [5]:
```
# Attributes of 3-dimensional array
```

In [6]:
```
# Import pandas and create a DataFrame out of one
# of the arrays you've created
```

In [7]:
```
# Create an array of shape (10, 2) with only ones
```

In [8]:
```
# Create an array of shape (7, 2, 3) of only zeros
```

In [9]:
```
# Create an array within a range of 0 and 100 with step 3
```

In [10]:
```
# Create a random array with numbers between 0 and 10 of size (7, 2)
```

In [11]:
```
# Create a random array of floats between 0 & 1 of shape (3, 5)
```

In [12]:
```
# Set the random seed to 42


# Create a random array of numbers between 0 & 10 of size (4, 6)
```

Run the cell above again, what happens?

Are the numbers in the array different or the same? Why do think this is?

In [13]:
```
# Create an array of random numbers between 1 & 10 of size (3, 7)
# and save it to a variable


# Find the unique numbers in the array you just created
```

In [14]:
```
# Find the 0'th index of the latest array you created
```

In [15]:
```
# Get the first 2 rows of latest array you created
```

In [16]:
```
# Get the first 2 values of the first 2 rows of the latest array
```

In [17]:
```
# Create a random array of numbers between 0 & 10 and an array of ones
# both of size (3, 5), save them both to variables
```

In [18]:
```
# Add the two arrays together
```

In [19]:
```
# Create another array of ones of shape (5, 3)
```

In [20]:
```
# Try add the array of ones and the other most recent array together
```

When you try the last cell, it produces an error. Why do think this is?

How would you fix it?

In [21]:
```
# Create another array of ones of shape (3, 5)
```

In [22]:
```
# Subtract the new array of ones from the other most recent array
```

In [23]:
```
# Multiply the ones array with the latest array
```

In [24]:
```
# Take the latest array to the power of 2 using '**'
```

In [25]:
```
# Do the same thing with np.square()
```

In [26]:
```
# Find the mean of the latest array using np.mean()
```

In [27]:
```
# Find the maximum of the latest array using np.max()
```

In [28]:
```
# Find the minimum of the latest array using np.min()
```

In [29]:
```
# Find the standard deviation of the latest array
```

In [30]:
```
# Find the variance of the latest array
```

In [31]:
```
# Reshape the latest array to (3, 5, 1)
```

In [32]:
```
# Transpose the latest array
```

What does the transpose do?

In [33]:
```
# Create two arrays of random integers between 0 to 10
# one of size (3, 3) the other of size (3, 2)
```

In [34]:
```
# Perform a dot product on the two newest arrays you created
```

In [35]:
```
# Create two arrays of random integers between 0 to 10
# both of size (4, 3)
```

In [36]:
```
# Perform a dot product on the two newest arrays you created
```

It doesn't work. How would you fix it?

In [37]:
```
# Take the latest two arrays, perform a transpose on one of them and then perform
# a dot product on them both
```

Notice how performing a transpose allows the dot product to happen.

Why is this?

Checking out the documentation on `np.dot()` may help, as well as reading Math is Fun's guide on the dot product.

Let's now compare arrays.

```
In [38]: # Create two arrays of random integers between 0 & 10 of the same shape
         # and save them to variables
```

```
In [39]: # Compare the two arrays with '>'
```

What happens when you compare the arrays with >?

```
In [40]: # Compare the two arrays with '>='
```

```
In [41]: # Find which elements of the first array are greater than 7
```

```
In [42]: # Which parts of each array are equal? (try using '==')
```

```
In [43]: # Sort one of the arrays you just created in ascending order
```

```
In [44]: # Sort the indexes of one of the arrays you just created
```

```
In [45]: # Find the index with the maximum value in one of the arrays you've created
```

```
In [46]: # Find the index with the minimum value in one of the arrays you've created
```

```
In [47]: # Find the indexes with the maximum values down the 1st axis (axis=1)
         # of one of the arrays you created
```

```
In [48]: # Find the indexes with the minimum values across the 0th axis (axis=0)
         # of one of the arrays you created
```

```
In [49]: # Create an array of normally distributed random numbers
```

```
In [50]: # Create an array with 10 evenly spaced numbers between 1 and 100
```