

✓ Numpy guide

Documentation and helpful links:

https://numpy.org/doc/stable/user/absolute_beginners.html

<https://www.w3schools.com/python/numpy/default.asp>

```
1 #first, import numpy
2 import numpy as np
```

✓ Creating and reading arrays

An array is a structure for storing and retrieving data. We often talk about an array as if it were a grid in space, with each cell storing one element of the data.

```
1 a = np.array([[1, 2, 3],
2               [4, 5, 6]])
3
4 shape = a.shape # tells the dimensions of the array
5 size = a.size # tells how many terms are in the array
6 index1 = a[0] # arrays can be treated like lists! we can index them!
7 index2 = a[0][0]
8
9 print(f"Our array is defined as {a} \n")
10 print(f"The shape of our array is {shape},\nwhere the first number refers to the number of rows and \nthe second to the number of columns.")
11 print(f"We can index a whole row using only one square bracket: {index1}")
12 print(f"To index a specific value in a row and column, we must use two square brackets: {index2}")
13
```

Our array is defined as `[[1 2 3]
[4 5 6]]`

The shape of our array is (2, 3),
where the first number refers to the number of rows and
the second to the number of columns.

We can index a whole row using only one square bracket: `[1 2 3]`
To index a specific value in a row and column, we must use two square brackets: `1`

```
1 b = np.array([20,30,40,50, 60, 70])
2
3 print(f"My original b array defined in line 1: {b}")
4 b[-1] = 10
5
6 print(f"Similarly to lists, arrays are mutable, as shown in line 4: {b}. \nThey can also be sliced: {b[:3]}")
```

My original b array defined in line 1: `[20 30 40 50 60 70]`
Similarly to lists, arrays are mutable, as shown in line 4: `[20 30 40 50 60 10]`.
They can also be sliced: `[20 30 40]`

Let's create a basic array:

```
1 # each number states how many elements you want in your array
2
3 only_zeros = np.zeros(5)
4 only_ones = np.ones(4)
5 empty = np.empty(3) # this creates an array whose initial content is random
6
7 range = np.arange(7) # this gives a range of elements
8 range2 = np.arange(3,15,3) # arange works similarly to lists
9 print(f"This is how we print arrays based on first number, \nlast number, and step size: {range2}\n")
10
11 linear = np.linspace(0,10, 5) # values spaced linearly in a specified interval: start, end, number of values inbetween
12 print(f"This is how we another way to create arrays with values \nspaced linearly: {linear}")
13
```

This is how we print arrays based on first number,
last number, and step size: `[ 3 6 9 12]`

This is how we another way to create arrays with values spaced linearly: [0. 2.5 5. 7.5 10.]

```
1 # you can also sort arrays
2 arr = np.array([2, 1, 5, 3, 7, 4, 6, 8])
3 np.sort(arr)
```

```
array([1, 2, 3, 4, 5, 6, 7, 8])
```

```
1 # what about concatenation?
2
3 a = np.array([1, 2, 3, 4])
4 b = np.array([5, 6, 7, 8])
5
6 np.concatenate((a, b))
```

```
array([1, 2, 3, 4, 5, 6, 7, 8])
```

Other array operations:

```
1 data = np.array([1, 2, 3, 4, 5, 6, 7, 8, 10, 20, 30, 40, 50, 60, 70, 80])
2
3 print(f"The maximum in this array is {data.max()}.")
4
5 print(f"The minimum in this array is {data.min()}.")
6
7 print(f"The sum of this array is {data.sum()}.")
8
9 print(f"The average of this array is {data.mean()}.")
10
11 print(f"The product of this array is {data.prod()}.")
```

```
The maximum in this array is 80.
The minimum in this array is 1.
The sum of this array is 396.
The average of this array is 24.75.
The product of this array is 162570240000000000.
```

```
1 list = [1, 2, 3, 4, 5, 6, 7, 8, 10, 20, 30, 40, 50, 60, 70, 80]
2
3 print(list)
4
5 array = np.array(list)
6
7 print(array)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 10, 20, 30, 40, 50, 60, 70, 80]
[ 1  2  3  4  5  6  7  8 10 20 30 40 50 60 70 80]
```

Application for Linear algebra

```
1 data = np.arange(12)
2
3 matrix = data.reshape(4,3) # row, column
4
5 print(data)
6 print(f"The original matrix is \n{matrix}")
7 print(f"The transposed matrix is \n{matrix.transpose()}")
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11]
The original matrix is
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
The transposed matrix is
[[ 0  3  6  9]
 [ 1  4  7 10]
 [ 2  5  8 11]]
```

