**Generating and Manipulating ndarrays**

NumPy has many *many* pages of documentation on all of its extensive functionality. But rather than go through the list one by one, the best way to actually learn NumPy is to apply it to a series of small problems. That way you can familiarise yourself with how to use NumPy for the common use cases that you'll encounter on your own data science journey too.

Challenge 1

Use [.arange()](https://numpy.org/devdocs/reference/generated/numpy.arange.html)to createa a vector a with values ranging from 10 to 29. You should get this:

print(a)

[10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29]

Challenge 2

Use Python slicing techniques on a to:

* Create an array containing only the last 3 values of a
* Create a subset with only the 4th, 5th, and 6th values
* Create a subset of a containing all the values except for the first 12 (i.e., [22, 23, 24, 25, 26, 27, 28, 29])
* Create a subset that only contains the even numbers (i.e, every second number)

Challenge 3

Reverse the order of the values in a, so that the first element comes last:

[29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10]

If you need a hint, you can check out this part of the [NumPy beginner's guide](https://numpy.org/devdocs/user/absolute_beginners.html#how-to-reverse-an-array)

Challenge 4

Print out all the indices of the non-zero elements in this array: [6,0,9,0,0,5,0]

Challenge 5

Use NumPy to generate a 3x3x3 array with random numbers

Hint: Use the [.random() function](https://numpy.org/doc/stable/reference/random/index.html?highlight=random#module-numpy.random)

Challenge 6

Use [.linspace()](https://numpy.org/doc/stable/reference/generated/numpy.linspace.html) to create a vector x of size 9 with values spaced out evenly between 0 to 100 (both included).

Challenge 7

Use [.linspace()](https://numpy.org/doc/stable/reference/generated/numpy.linspace.html) to create another vector y of size 9 with values between -3 to 3 (both included). Then plot x and y on a line chart using Matplotlib.

Challenge 8

Use NumPy to generate an array called noise with shape 128x128x3 that has random values. Then use Matplotlib's [.imshow()](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.imshow.html) to display the array as an image.

The random values will be interpreted as the RGB colours for each pixel.

.

.

..

...

..

.

.

**Solution: To the NumPy Mini Challenges**

**Challenge 1:** Previously we created NumPy arrays manually where we specified each and every value like this: np.array([1.1, 9.2, 8.1, 4.7])

We can also generate a NumPy arrays using some built-in functions like .arange(). In this case, we can create an array of evenly spaced values by just providing a start and stop value.

1. a = np.arange(10,30)
2. print(a)

**Challenge 2:** This should be a little bit of revision for using the colon : operator to select a range or interval in an array.

The last 3 values in the array:

1. a[-3:]

An interval between two values:

1. a[3:6]

All the values except the first 12:

1. a[12:]

Every second value (i.e., all the even values in our case)

1. a[::2]

**Challenge 3:**To reverse the order of an array, you can either use the (double) colon operator once again or use the built-in .flip() function. Either way works.

1. np.flip(a)

or

1. a[::-1]

**Challenge 4:** If you did a quick Google search, chances are you discovered the built-in .nonzero() function to print out all the non-zero elements. You can use it like so:

1. b = np.array([6,0,9,0,0,5,0])
2. nz\_indices = np.nonzero(b)
3. nz\_indices # note this is a tuple

**Challenge 5:** The .random() function is another way to quickly create a ndarray, just like .arange(). The .random() function lives under np.random so you'll either have to import random

1. from numpy.random import random
2. z = random((3,3,3))
3. z

or use the full path to call it.

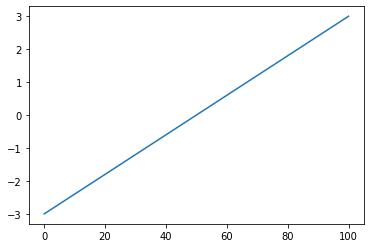
1. z = np.random.random((3,3,3)) # without an import statement
2. print(z.shape)
3. z

**Challenge 6**: The .linspace() function is very similar to .arange() and great for generating evenly spaced numbers over an interval. To generate the vector use:

1. x = np.linspace(0, 100, num=9)
2. print(x)
3. x.shape

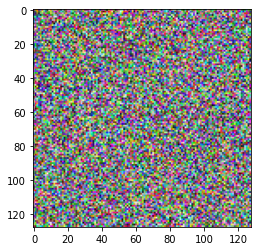
**Challenge 7:**A common use-case for .linspace() is to generate the points that you'd like to plot on a chart.

1. y = np.linspace(start=-3, stop=3, num=9)
2. plt.plot(x, y)
3. plt.show()



**Challenge 8:** When you have a 3-dimensional array with values between 0 and 1, we can use Matplotlib to interpret these values as the red-green-blue (RGB) values for a pixel.

1. noise = np.random.random((128,128,3))
2. print(noise.shape)
3. plt.imshow(noise)



That's pretty cool, right?! We've just generated a 128x128 pixel image of random noise because each dimension in our NumPy array can be interpreted to hold the colour information for a pixel.