Numpy review homework

```
In [58]: import numpy as np
```

1. Make a numpy matrix from a Python list of lists...

```
In [59]: # python list with nested list
         python_list = [[2, 4, 6], [8, 10, 12], [14, 16, 18]]

         # python list to numpy array
         py_to_np = np.array(python_list)
         py_to_np
```

```
Out[59]: array([[ 2,  4,  6],
                [ 8, 10, 12],
                [14, 16, 18]])
```

2. Make a 3D numpy matrix from a Python list of lists of lists!

```
In [60]: new_list = [[[1, 2], [3, 4], [5, 6]], [[7, 8], [9, 10], [11, 12]]]

         array_3D = np.array(new_list)
         array_3D
```

```
Out[60]: array([[[ 1,  2],
                 [ 3,  4],
                 [ 5,  6]],

                [[ 7,  8],
                 [ 9, 10],
                 [11, 12]]])
```

3. Create a 5x3 array of Gaussian random numbers.

```
In [63]: n_rows = 5
         n_columns = 3

         rand_array = np.random.randn(n_rows, n_columns)
         rand_array
```

```
Out[63]: array([[ 2.75601012,  0.8067011 ,  0.51408022],
                [-2.59083947, -2.30813194,  0.52096573],
                [ 0.65137241,  0.10573102,  0.51720422],
                [ 0.44135834, -0.07414974,  0.96706359],
                [-0.71996156, -0.71319706,  0.06748372]])
```

4. Write a script to go through the array created in 3. and announce (print) the value and its row and column indexes.

Hint: Use nested `for` loops - one to loop through the rows and one to loop through the columns.

```
In [72]: for i in range(n_rows):
             for j in range(n_columns):
                 value = rand_array[i][j]
```

```
        row_index = i
        column_index = j
        print(f"Value: {value:.2f}, row index: {row_index}, column index: {
```

```
Value: 2.76, row index: 0, column index: 0.
Value: 0.81, row index: 0, column index: 1.
Value: 0.51, row index: 0, column index: 2.
Value: -2.59, row index: 1, column index: 0.
Value: -2.31, row index: 1, column index: 1.
Value: 0.52, row index: 1, column index: 2.
Value: 0.65, row index: 2, column index: 0.
Value: 0.11, row index: 2, column index: 1.
Value: 0.52, row index: 2, column index: 2.
Value: 0.44, row index: 3, column index: 0.
Value: -0.07, row index: 3, column index: 1.
Value: 0.97, row index: 3, column index: 2.
Value: -0.72, row index: 4, column index: 0.
Value: -0.71, row index: 4, column index: 1.
Value: 0.07, row index: 4, column index: 2.
```

5. Make an new array out of your random numbers such that the mean is 10 and the standard deviation is 3.

In [74]:

```
mean = 10
sd = 3

new_rand_array = (rand_array * sd) + mean
new_rand_array
```

Out[74]:

```
array([[18.26803036, 12.42010329, 11.54224065],
       [ 2.22748159,  3.07560419, 11.5628972 ],
       [11.95411723, 10.31719307, 11.55161267],
       [11.32407502,  9.77755079, 12.90119076],
       [ 7.84011531,  7.86040881, 10.20245116]])
```

6. Count the number of values in your new array that are below 7.

In [77]:

```
count = new_rand_array < 7

# count.sum() counts the number of True values

print(f"There are {count.sum()} values in the array that are below 7.")
```

```
There are 2 values in the array that are below 7.
```

7. Make a numpy sequence that has the even numbers from 2 up to (and including) 20.

In [78]:

```
even_array = np.arange(2, 21, 2)
even_array
```

Out[78]:

```
array([ 2,  4,  6,  8, 10, 12, 14, 16, 18, 20])
```

8. Get the second and third rows of your array created in #5.

In [79]:

```
new_rand_array[1:3, :]
```

```
Out[79]: array([[ 2.22748159,  3.07560419, 11.5628972 ],
                [11.95411723, 10.31719307, 11.55161267]])
```

9. Compute the mean of the columns of your array created in #5.

```
In [80]: new_rand_array.mean(0) # mean(0) is means of columns
```

```
Out[80]: array([10.3227639 ,  8.69017203, 11.55207849])
```