Exercise 1

Check if all elements from the following arrays return the logical value *True*:

```
A = np.array([[3, 2, 1, 4],
              [5, 2, 1, 6]])
B = np.array([[3, 2, 1, 4],
              [5, 2, 0, 6]])
C = np.array([[True, False, False],
              [True, True, True]])
D = np.array([0.1, 0.3])
```

Print result to the console as shown below.

Tip: Use the function np.all().

Expected result:

```
A: True
B: False
C: False
D: True
```

Exercise 2

Using *Numpy* create a one-dimensional array of all two-digit numbers and print this array to the console as shown below.

Tip: Use the np.arange() function.

Expected result:

```
[10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57
58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81
82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99]
```

Exercise 3

Using *Numpy* create the following array:

```
[[10 11 12 13 14 15 16 17 18 19]
[20 21 22 23 24 25 26 27 28 29]
[30 31 32 33 34 35 36 37 38 39]
[40 41 42 43 44 45 46 47 48 49]
[50 51 52 53 54 55 56 57 58 59]
[60 61 62 63 64 65 66 67 68 69]
[70 71 72 73 74 75 76 77 78 79]
[80 81 82 83 84 85 86 87 88 89]
[90 91 92 93 94 95 96 97 98 99]]
```

Note that the shape of this array is (9, 10). In response, print array to the console.

Tip: Use the np.arange() function and the np.ndarray.reshape() method.

Exercise 4
Using *Numpy* create a one-dimensional array (vector) containing the possible result from the *Big Lotto* game. Set the random seed to 42. Print result to the console.

Tip: The result of this game is a 6-element vector of values from 1 to 49 inclusive.

Expected result:

```
[14 46 48 45 18 28]
```

Exercise 5

Using *Numpy* create the following two-dimensional array:

```
[[0 0 0 0 0 0]
[0 1 0 0 0 0]
[0 0 2 0 0 0]
[0 0 0 3 0 0]
[0 0 0 0 4 0]
[0 0 0 0 0 5]]
```

Print result to the console as shown below.

Tip: Use the np.diag() function

Exercise 6
Using *Numpy* create the following array:

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

Save this array to a binary file named 'array.npy'and then load that file back into another variable. Print this variable to the console.

Tip: Use the np.save()and np.load()functions.

Exercise 7

Using *pandas,* from the list below:

```
stocks = ['PLW', 'CDR', '11B', 'TEN']
```

create a *Series* object and print it to the console.

Expected result:

```
0    PLW
1    CDR
2    11B
3    TEN
dtype: object
```

Exercise 8

Using *pandas*, from the dictionary below:

```
stocks = {'PLW': 387.00, 'CDR': 339.5, 'TEN': 349.5, '11B': 391.0}
```

create a *Series* object and assign it to the *quotations* variable. In response, print *quotations* variable to the console.

Expected result:

```
PLW    387.0
CDR    339.5
TEN    349.5
11B    391.0
dtype: float64
```

Exercise 9

The following *Series* is given (*quotations* variable):

```
PLW    387.0
CDR    339.5
TEN    349.5
11B    391.0
dtype: float64
```

Convert *quotations* to the list and print it to the console.

Expected result:

```
[387.0, 339.5, 349.5, 391.0]
```

Exercise 10

The following *Series* is given:

```
series = pd.Series(['001', '002', '003', '004'], list('abcd'))
```

Convert its type to *int* and print this *Series* to the console.

Expected result:

```
a    1
b    2
c    3
d    4
dtype: int64
```

Exercise 11

The *df DataFrame* is given below. Extract rows from this *DataFrame* for which the *col2* column is between 0.0 and 1.0 (inclusive).
In response, print result to the console.

Expected result:

```
1.        col1      col2
2.1   0.950714  0.314247
3.6   0.058084  0.067528
4.9   0.708073  0.110923
5.11  0.969910  0.375698
6.18  0.431945  0.822545
```

Exercise 12

The *df DataFrame* is given below. Calculate the median for the *col2* and print it to the console.

Expected result:

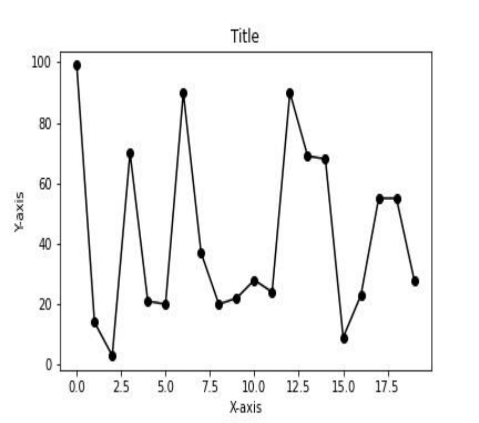
```
1.-0.4180382371592297
```

Exercise 13

Use the following arrays to build a line graph. Within the plt.plot() function change the marker to 'o', (marker = 'o') and the line color to 'black' (color = 'black').

```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(20)
y = np.random.randint(1,100, 20)
```

Expected result:



Exercise 14
With the data below build a histogram. Use the plt.hist() function.

```
import numpy as np
data = np.random.normal(loc = 100, scale = 0.5, size = 100)
```

Exercise 15

Use the data below to build the chart below. Change the font size to 15 on the axes and 20 on the title. Use the plt.legend() function to insert a legend on the chart, use the loc argument to set the position. Possibilities for setting the loc argument: best, upper right, upper left, lower left, lower right, right, center left, center right, lower center, upper center, center.

```
import numpy as np
import matplotlib.pyplot as plt

x = np.arange(1,11)
y1 = x**2
y2 = x**3
y3 = x**4
```

Exercise 16

Replicate the graph below and use the plt.savefig () function to save the result.

```
brand_A = [120, 130, 145, 177, 270, 211]
brand_B = [90, 41, 140, 150, 230, 193]
months = ['January','February','March','April','May','June']
```