Exercise 1: make a common array

```
In [ ]: import numpy as np

        # Seed insures results are stable.
        np.random.seed(21)
        random_integers = np.random.randint(1, high=500000, size=(20, 5))
        random_integers
```

```
Out[ ]: array([[ 80842, 333008, 202553, 140037,  81969],
               [ 63857,  42105, 261540, 481981, 176739],
               [489984, 326386, 110795, 394863,  25024],
               [ 38317,  49982, 408830, 485118,  16119],
               [407675, 231729, 265455, 109413, 103399],
               [174677, 343356, 301717, 224120, 401101],
               [140473, 254634, 112262,  25063, 108262],
               [375059, 406983, 208947, 115641, 296685],
               [444899, 129585, 171318, 313094, 425041],
               [188411, 335140, 141681,  59641, 211420],
               [287650,   8973, 477425, 382803, 465168],
               [  3975,  32213, 160603, 275485, 388234],
               [246225,  56174, 244097,   9350, 496966],
               [225516, 273338,  73335, 283013, 212813],
               [ 38175, 282399, 318413, 337639, 379802],
               [198049, 101115, 419547, 260219, 325793],
               [148593, 425024, 348570, 117968, 107007],
               [ 52547, 180346, 178760, 305186, 262153],
               [ 11835, 449971, 494184, 472031, 353049],
               [476442,  35455, 191553, 384154,  29917]])
```

Exercise 2:What is the average value of the second column (to one decimal place)

```
In [ ]: # The average value of the second column
        random_integers[:, 1].mean()
```

```
Out[ ]: 214895.8
```

Exercise 3: What is the average value of the first 5 rows of the third and fourth columns (to one decimal place)?

```
In [ ]: # The average of the first 5 rows of 3rd and 4th columns
        subset = random_integers[:5, 2:4]
        np.mean(subset)
```

```
Out[ ]: 286058.5
```

Exercise 4: Result of matrix 1 plus matrix 2

$$\begin{bmatrix} 2 & 4 & 6 \\ 5 & 7 & 9 \end{bmatrix}$$

In [ ]:
```
# Exercise 4 Python:
first_matrix = np.array([[1, 2, 3], [4, 5, 6]])
second_matrix = np.array([1, 2, 3])
print(first_matrix + second_matrix)
```

```
[[2 4 6]
 [5 7 9]]
```

Exercise 5: Result of my_vector[selection]:

$$\begin{bmatrix} 2 & 4 & 6 \end{bmatrix}$$

In [ ]:
```
# Exercise 5 python:
my_vector = np.array([1, 2, 3, 4, 5, 6])
selection = my_vector % 2 == 0
print(my_vector[selection])
```

```
[2 4 6]
```

For exercise 6: I didn't make any errors but I learned how to do matrix notation on markdown

Exercise 7 slicing:

$$\begin{bmatrix} 2 & 3 \\ 5 & 6 \end{bmatrix}$$

In [ ]:
```
# Exercise 8
my_array = np.array([[1, 2, 3], [4, 5, 6]])
my_slice = my_array[:, 1:3]
my_array[:, :] = my_array * 2
print(my_slice)
```

```
[[ 4  6]
 [10 12]]
```

Exercise 8 slicing and view

$$\begin{bmatrix} 4 & 6 \\ 10 & 12 \end{bmatrix}$$

Exercise 9 what does the slice look like?

$$\begin{bmatrix} 2 & 3 \\ 5 & 6 \end{bmatrix}$$

Exercise 10: my prediction was correct. I knew that slice would not change because my_slice creates its own subsetted array that is different from the original my array

In [ ]:
```
# exercise 10
my_array = np.array([[1, 2, 3], [4, 5, 6]])
my_slice = my_array[:, 1:3]
```

```
my_array = my_array * 2
print(my_slice)
```

```
[[2 3]
 [5 6]]
```

Exercise 11:

$$\begin{bmatrix} 2 & 3 \\ 5 & 6 \end{bmatrix}$$

In [ ]:
```
my_array = np.array([[1, 2, 3], [4, 5, 6]])
my_slice = my_array[:, 1:3].copy()
my_array[:, :] = my_array * 2
print(my_slice)
```

```
[[2 3]
 [5 6]]
```

Exercise 12 prediction: y would be ["a change", 2]

Exercise 13 prediction: if we printed x it would be [1,2,3]

In [ ]:
```
# Exercise 12 and 13:
x = [1, 2, 3]
y = x[0:2]
y[0] = "a change"
print(y)
print(x)
```

```
['a change', 2]
[1, 2, 3]
```

Exercise 14: I was correct and this is because we sliced y and then made a change on a specific index number for y and also x doesn't change at all.

In [ ]:
```
my_array = np.array([1, 2, 3])
my_array = my_array[1:4]
print(my_array)
my_slice = my_array[1:3]
print(my_slice)
my_slice[0] = -1
print(my_array)
print(my_slice)
```

```
[2 3]
[3]
[ 2 -1]
[-1]
```