Kian Bagherlee Numpy Submission

Exercise 1

```
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
```

```
c:\Users\kbagh\miniconda3\Lib\site-packages\numpy\_distributor_init.py:30: UserWarni
ng: loaded more than 1 DLL from .libs:
c:\Users\kbagh\miniconda3\Lib\site-packages\numpy\.libs\libopenblas64__v0.3.21-gcc_1
0_3_0.dll
c:\Users\kbagh\miniconda3\Lib\site-packages\numpy\.libs\libopenblas64__v0.3.23-246-g
3d31191b-gcc_10_3_0.dll
  warnings.warn("loaded more than 1 DLL from .libs:"
```

```
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

```
In [ ]: np.mean(random_integers, axis=0)[1]
```

```
Out[ ]: 214895.8
```

Exercise 3

```
In [ ]: np.mean(random_integers[0:5, 3:])
```

```
Out[ ]: 201466.2
```

Exercise 4

I think that the result of this will be adding second_matrix to each row of list in first_matrix. So the result will be:

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

Exercise 5

$$\begin{bmatrix} False & True & False & True & False & True \end{bmatrix}$$

Exercise 6

```
In [ ]: 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]]
```

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

```
[2 4 6]
```

My answer for the first question was correct, where each row of the matrix had its values added. My answer for the second question was incorrect, where I thought that the result of the code would be an array that held whether each value properly followed the stated rule. In actuality, selection just holds the actual values from the matrix that were True, instead of the value True itself. The mistake was that I misread the print statement

Exercise 7

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

Exercise 8

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

Exercise 9

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

Exercise 10

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

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

In [ ]:
```
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]]
```

In [ ]:
```
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]]
```

My predictions were the other way around. I initially thought that my_array[:,:] was the format that you use when you want to make changes to an array, but have it as a copy, not a view. I misinterpreted it, and now realize that utilizing [:,:] on the left-hand side is how you verify that it's used as a view and not a copy, as it sustains the link to the original array.

Exercise 11

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

Exercise 12

["a change", 2]

Exercise 13

[1, 2, 3]

Exercise 14

```
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]]
```

```
In [ ]: x = [1, 2, 3]
        y = x[0:2]
        y[0] = "a change"
        print(y)
```

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

```
In [ ]: print(x)
```

```
[1, 2, 3]
```

I was correct with all of my guesses. I was correct about my first guess because I recognized that there was a .copy() being used, and I was correct about the next two guesses because I recognized that we were working with python lists.