Numpy Exercises - due tuesday 09/09 | *by Larissa and Charlotte*

Exercise 1

First, lets make a common array to work with.

In [1]:
```
import numpy as np
np.random.seed(21) # This guarantees the code will generate the same set o
random_integers = np.random.randint(1,high=500000, size=(20, 5))
random_integers
```

Out[1]:
```
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 two decimal places)

In [2]:
```
np.mean(random_integers[:,1])
```

Out[2]:
```
214895.8
```

Exercise 3

What is the average value of the first 5 rows of the third and fourth columns?

In [3]:
```
np.mean(random_integers[0:5, 2:4])
```

Out[3]:
```
286058.5
```

Exercise 4

Close Python. On a piece of paper, write down the final result of the following code:

import numpy as np
first_matrix = np.array([[1, 2, 3], [4, 5, 6]])
print(first_matrix)
second_matrix = np.array([1, 2, 3])
print(second_matrix)
first_matrix + second_matrix

- Result should be:
  [2 4 6]
  [4 5 6]

Exercise 5

Keep Python **Closed!** Write down the final result of the following code:

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

- Result should be:
  [2 4 6]

Exercise 6

Now open python and check your answers to Exercises 4 and 5.

```
In [6]:  first_matrix + second_matrix
```

```
Out[6]:  array([[2, 4, 6],
                [5, 7, 9]])
```

Incorrect. But the result does indeed make sense, because the second matrix is added to both rows of the first_matrix.

```
In [7]:  my_vector = np.array([1, 2, 3, 4, 5, 6])
         selection = my_vector % 2 == 0
         my_vector[selection]
```

```
Out[7]:  array([2, 4, 6])
```

Correct assumption.

Exercise 7

Close your computer / laptop. Let's try and work out a few problems in our heads to test our understanding of numpy views. Let's start with the following array:

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

[[1 2 3]
[4 5 6]]

Now, on a piece of paper write down the value of my_slice = my_array[:, 1:3].

- Result should be:
  [2 3]
  [5 6]

Exercise 8

Now suppose we run the code my_array[:, :] = my_array * 2. Now what does my_slice look like?

- Result should be:
  [4 6]
  [10 12]
  ... because we made a view and not a copy, i.e. changes in my_array will be reflected in my_slice too.

Exercise 9

Now suppose we run my_array = my_array * 2. What does my_slice look like?

- Result should still be:
  [4 6]
  [10 12]
  ... because the same applies as in exercise 8.

Exercise 10

Stop, open Python, and try running these examples. Were your predictions correct? If not, why not?

In [14]:
```
my_array = np.array([[1, 2, 3], [4, 5, 6]])
print(my_array)
```

```
[[1 2 3]
 [4 5 6]]
```

In [15]:
```
my_slice = my_array[:, 1:3]
print(my_slice)
```

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

Correct!

In [16]:
```
my_array[:, :] = my_array * 2
print(my_slice)
```

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

Correct!

In [17]:
```
my_array = my_array * 2
print(my_slice)
```

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

Correct!

Exercise 11

OK, let's close Python again and go back to pen and paper. Let's also reset my_array and start over with the following code:

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

my_slice = my_array[:, 1:3].copy()
print(my_slice)
[[2 3]
[5 6]]

Now suppose we run the following code: my_array[:, :] = my_array * 2. What does my_slice look like?

- Result should be:
  [2 3]
  [5 6]
  ...because we created a copy and not a view. When we change my_array this change will not be reflected in my_slice because they point to two different lists.

In [18]:
```
my_array = np.array([[1, 2, 3], [4, 5, 6]])
print(my_array)
my_slice = my_array[:, 1:3].copy()
print(my_slice)
my_array[:, :] = my_array * 2
print(my_slice)
```

```
[[1 2 3]
 [4 5 6]]
[[2 3]
 [5 6]]
[[2 3]
 [5 6]]
```

Assumption was correct!