

KEY_Lesson13_Numpy_Intro

July 15, 2019

1 Introduction to Numpy

In the last two days of camp, you have learned how to start using python and how to manipulate data with the pandas library. Today, we will be learning how to perform calculations with another library known as numpy. numpy allows you to do math on entire lists of data, all at the same time!

To review what we learned on monday, let's create a list containing the numbers 0 through 5 and assign it to a variable:

```
[0]: # Create a list containing the numbers 0 through 4
data = [0, 1, 2, 3, 4]
```

```
# or convert the range to a list
data = list(range(5))
```

```
[2]: # let's see what it looks like
print(data)
```

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

Using lists can be really useful, because you can store any set of data in the. For instance, you could store the temperature in your house each morning as a list.

Now, lets add 1 to each of the items in the list. Is there a simple way to do this?

```
[3]: data + 1
```

```

      □
↳ -----
```

```

      TypeError                                Traceback (most recent call↳
↳ last)
```

```

      <ipython-input-3-1943d790f594> in <module>()
----> 1 data + 1
```

```

      TypeError: can only concatenate list (not "int") to list
```

hmm ... that didn't work! That is because the + operator acts as a concatenation operator on lists, and we learned in Lesson 2 that we can only concatenate lists with lists, not integers.

Another way we could update our list is to add 1 to each of the items in the list individually. We can do this by **indexing** to isolate each value in the list one by one.

```
[0]: # Add 1 to each item in the array you created above
data[0] = data[0] + 1
data[1] = data[1] + 1
data[2] = data[2] + 1
data[3] = data[3] + 1
data[4] = data[4] + 1

# print the list to see how it changed
print(data)
```

You might have noticed that this is a little less convenient than with a single variable, where you can do something like this to add a number:

```
[0]: # create a single variable and add a number to it
a = 5
a += 1
print(a)
```

Today we will be using numpy, which allows us to quickly and efficiently perform mathematical operations on entire lists (or, as they're called in numpy, *arrays*)!

First we will import numpy. Remember when we imported pandas, we gave it the special nickname pd? We're also going to give numpy a nickname: np:

```
[0]: # Load numpy
import numpy as np
```

Now whenever we type np, python will know we really mean numpy.

There is a ton of useful stuff we can do with numpy, so let's re-do the example above using numpy arrays instead of lists.

```
[0]: # create a numpy array containing the numbers 0 through 4
data_array = np.array([0, 1, 2, 3, 4])

# you can print arrays just like lists
print(data_array)
```

Numpy arrays act very similarly to regular python lists, we can grab individual items using the same syntax:

```
[0]: # Print any number from the numpy array you just created:
print(data_array[0])
```

Numpy arrays also add a lot of useful features such as the ability to perform commands *on all items in a list at the same time*.

We can demonstrate this by adding a number to all of the items in the Numpy array:

```
[0]: # Add any number to the array we created above
data_array += 5
```

```
# print the array  
print(data_array)
```

See how much easier that was than manually changing each element of a list? We will be using numpy for the rest of the day to perform calculations on arrays of data. In the above example we used addition, but you can also perform any mathematical operation we've talked about with numpy arrays.

In this lesson you learned how to: * Load numpy into Python. * Create an array with numpy. * Perform math with numpy arrays.

Now, let's continue to practice with your partner!