Numpy and Pandas for Data Analysis

In this tutorial, I present a detailed overview of the use of Numpy and Pandas for data analysis.

Numpy - "Numerical Python"

Numpy is the mainstream package for numerical computing in Python. It uses C -API. This means data are passed to external libraries like C. They then return data back to Python as Numpy arrays.

Advantages:

- Efficiency on large collections of data. (eg. a list of 1milionl item)
- Performs operations without the explicit need for loops.

Importing Numpy

To use numpy you need to first import it using the command below.

```
import numpy as np
```

Ndarray - n dimensional array object

Overview. As an overview of Numpy array. I have created a 2 -dimensional array with dimensions (2 X 3). All codes are in the color turquoise

```
data = np.random.randn(2, 3)

data

data * 10

data + data
```

Properties:

- All elements have the same type

```
data.dtype
```

- Every array has a shape (tuple indicating the size of each dimension)

```
data.shape
```

Creating ndarrays

Can be created from a list. See below

```
data1 = [6, 7.5, 8, 0, 1] # a list
arr1 = np.array(data1) # an array from the list
```

can also create from a multidimensional list

```
data2 = [[1, 2, 3, 4], [5, 6, 7, 8]]
arr2 = np.array(data2)
arr2
```

Check shape. A shape is sort of the size.

```
arr2.shape
```

Check dimension

```
arr2.ndim
```

np.array infers data type based on the input. If all your inputs are integers. Then np.array infers that automatically.

```
arr1.dtype
arr2.dtype
```

Create ones, zeros, empty and a range of values

```
np.zeros(10)
np.ones(10)
np.arange(12)
```

```
np.zeros((2,3))
```

Data type:

```
arr1 = np.array([1, 2, 3], dtype=np.float64)
arr2 = np.array([1, 2, 3], dtype=np.int32)
arr1.dtype
arr2.dtype
```

Arithmetic with arrays

```
arr = np.array([[1., 2., 3.], [4., 5., 6.]])
arr *arr
arr - arr
1/arr (1 divides each element)

arr2 = np.array([[0., 4., 1.], [7., 2., 12.]])
arr2 > arr
```

Indexing and slicing

It is good to note that a one-dimensional array act like a Python list. For example:

```
arr = np.arange(10)
arr
arr[5]
arr[5:8]
arr[5:8] = 12
```

*note ** (array slices are views. data are not copied. any changes affect source data.)*

```
arr_slice = arr[5:8]

arr_slice[1] = 12345

arr
```

Two-dimensional array.

They are different from one-dimensional arrays, especially in the way they are indexed and sliced. elements at each index correspond to an array. In order to select a single element from a 2D array. The first index corresponds to the first axis while the second index extracts the single element needed.

An example is illustrated below

```
arr2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]) # create the array.

arr2d[2] # extract third element which is in fact third array from the arr2d defined

above array([7, 8, 9])

arr2d[0][2]

arr2d[0,2]
```

More slices

```
arr = array([ 0, 1, 2, 3, 4, 64, 64, 64, 8, 9])

arr[1:6]

arr2d

arr2d[:2]

arr2d[:2,1:]
```

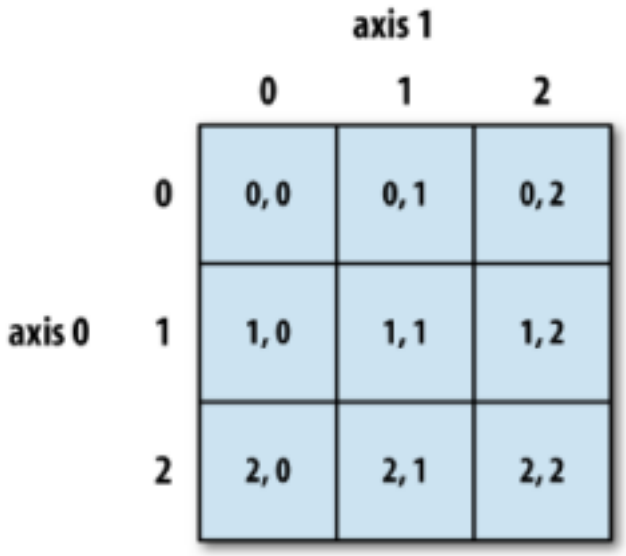
Select the second row but only the first two columns like :

```
arr2d[1, :2]
```

Similarly, I can select the third column but only the first two rows like:

```
arr2d[:2, 2]
```

2d array Indexing representation.



Sorting

Numpy has an input sorting function demonstrated below.

```
arr = np.random.randn(6)
arr
arr.sort()
```

Unique values

You can extract the unique values from a numpy array that has repetitions.

```
names = np.array(['Bob', 'Joe', 'Will', 'Bob', 'Will', 'Joe', 'Joe'])
```

```
np.unique(names)
```

Read and write

Save and reload numpy arrays. This does not directly write to file! I will teach you how to use Pandas to write to file

```
arr = np.arange(10)
np.save('some_array', arr)
np.load('some_array.npy')
```

Pandas:

Just as we said about numpy. Pandas is another Python library built to be efficient and fast. It is being used extensively for data analysis. It uses the Data Frame structure and concept. You can think of a data frame as a table with columns and rows.

```
import pandas as pd
```

Create an empty data frame

```
df = pd.DataFrame()
print(df)
```

Create a data frame from a list

```
num_list = [1,2,3,4,5]
df = pd.DataFrame(num_list)
print(df)
```

Create from dictionary

Another effective way to create a data frame is to use a dictionary. A dictionary is a very efficient data structure, it contains a collection of key-value pairs. To create a data frame from a dictionary. The key becomes the column name and each value of the dictionary which is a list becomes the column values. An example is shown below.

Import from csv

Here is an example to import files using Pandas. I have an excel file here that I would share with the class during the next class day. As bro tuned taught on files. You need to put th file in the same directory as the project in PyCharm. That way the code below will run. Otherwise it would give you errors.

```
df= pd.read_csv('person.csv')
print(df)
```

Print the first few lines

```
df.head()
```

Print last few lines

```
df.tail()
```

Extract column.

Two ways.

```
df.name or df[‘name’]
```
#supposing we want to extract columns names. Each columns of a Pandas data frame is called a Pandas series.

Add column

```
df['HasBike'] = False
df.head()
```

drop column.

```
df.drop('HasBike', inplace=True, axis=1)
df.head()
```

save

```
df.to_csv('filename.csv')
```

Exercise

To consolidate this knowledge, kindly practice the code in this documentation. The file to use to practice the pandas code can be found downloaded with the link below.

https://drive.google.com/open?id=1rUxlDKvgVvx5hCBDEOYsVuu61_iM4zf0