Workshop 1

Learning From Big Data

Part 1: Python and Jupyter notebook

- From now on, we will have "lab work" associated with the topics lectured on each session. This is very important part of learning in the course, dealing with considerably practical work, such as programming, samples, exercises, tutorials and some additional topics not given in lecture time.

- During each workshop, you may follow the instructions or guides, given in the lab class, to carry out the exercises by yourself. However, I will give some key points and explanations, depending the topics and questions common among you. Please feel free to ask me any questions in the class, I will be happy to assist you.

- If some tasks or exercises are marked as "**Optional**", it means that you don't have to complete them in the class, depending on your interest or time.

Task 1: Practice with Jupyter Notebook

- As introduced in my lecture, Jupyter notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Its uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

  1). As you have installed jupyter notebook on your Mac or Ubuntu Virtual Box, usually start it in **your home directory** at first:

  ```
  $cd
  $ jupyter notebook
  …..
  ```

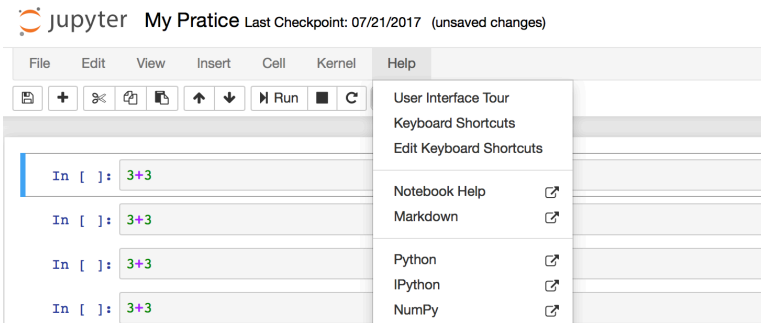
  2). Then, following the instructions from,

  http://nbviewer.jupyter.org/github/ipython/ipython/blob/3.x/examples/Notebook/Index.ipynb, to learn its basic topics as below (we temporarily leave the others in later when you need them)"

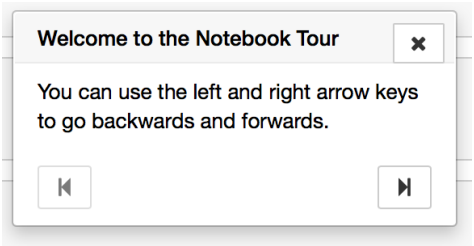
  ### Tutorials

  - What is the IPython Notebook
  - Notebook Basics
  - Running Code
  - Working With Markdown Cells

  3). A live explanation of the user interface of the notebook is given on "help" as shown as



- just simply click "User Interface Tour", you may get a prompt as like this:



- Then, you can go ahead by clicking "forward" icon for learning its user interface.

4). See how IPython and Jupyter Notebook work and their difference, from this page:

http://jupyter.readthedocs.io/en/latest/architecture/how_jupyter_ipython_work.html

Task 2: Learn Python Basics with Jupyter notebook

- Each time when you start your Jupyter notebook, it's better to do it at your home directory, as below:

```
$ cd
$ jupyter notebook
```

- Use your web-browser to navigate to

  .../2019 Big Data/Labs/Lab_1/Python_Basics_Notebook

- Have a practice with this notebook to learn Python basics.

- Find ipynb file: My_Practice.ipynb, where you may play any editing or running Python programs around!

Part 2: Spark *DataFrame*, *Pandas* and *Numy* with Python

Working on big data, we need to *transform* various kinds of data sources into a certain format or structure, so as to meet your project requirement. In recent years, a number of technologies have been developed and popularly used for achieving this purpose.

In this workshop associated with the topics lectured, you will have some practical works on such kind of technologies, *particularly in Python*.

Task 1: Spark DataFrames and SQL

- Spark DataFrames and SQL API is provided for connecting to **any** data source in the **same** way. It is essential for us to understand the API. By using the API, specially its modules, classes or functions, you may develop applications in Python.
- It is not necessary to study all of them at moment. Instead, you only know its big picture and part of them used in our samples at this stage.
- Nobody can memorize all of them, but one able to look up them, whenever you need.

Actions:

- Have a look at this link, pyspark.sql module , understand its big picture of the API at first.

Task 2: Review the notebooks given in Lab1

In the folder .../Labs/Lab1, I demonstrated a number of Ipython documents (.ipynb files) for lecturing the topic of "Chapter 3: Spark DataFrame and SQL" as below:

- Creating Spark DataFrame and Using SQL.ipynb
- DataFrame_Basic_Operations.ipynb
- GroupBy_and_Aggregate.ipynb
- Missing_Data.ipynb",
- Dates_and_Timestamps.ipynb

Actions:

1) Start *Jupyter notebook* at your **home directory** and then navigate to .../2019 Big Data/Labs/Lab1/Spark_DataFrame, where you will find them.

2) Review each notebook. You may have a practice for editing and running these samples and to try your idea or coding as you wish.

3) Referring to these samples, learn how to use Spark DataFrame and SQL API.

Some Helpful Notes for looking at the syntax of the APIs:

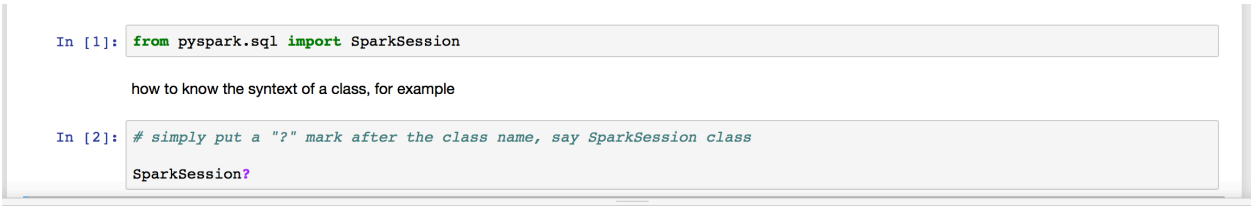
Way 1:

Using **"?"** (or **help** command) to see the specification or syntax of a class or function defined in their API documents, for examples:

- See the specification of class SparkSession

  or type

  help(SparkSession)



```
In [1]: from pyspark.sql import SparkSession

        how to know the syntext of a class, for example

In [2]: # simply put a "?" mark after the class name, say SparkSession class

        SparkSession?
```

```
Init signature: SparkSession(sparkContext, jsparkSession=None)
Docstring:
The entry point to programming Spark with the Dataset and DataFrame API.

A SparkSession can be used create :class:`DataFrame`, register :class:`DataFrame` as
tables, execute SQL over tables, cache tables, and read parquet files.
To create a SparkSession, use the following builder pattern:

>>> spark = SparkSession.builder \
...     .master("local") \
...     .appName("Word Count") \
...     .config("spark.some.config.option", "some-value") \
...     .getOrCreate()

.. autoattribute:: builder
   :annotation:
Init docstring:
Creates a new SparkSession.
```

- See the syntax (signature) of function spark.read.csv

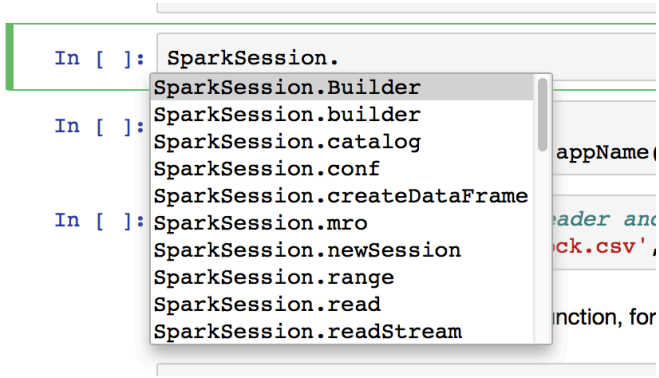
```
In [4]: # Let Spark know about the header and infer the Schema types!
        df = spark.read.csv('appl_stock.csv',inferSchema=True,header=True)

        Similarly, you can get the syntax of a function, for example for read()

In [6]: # see the specification or signature of a function spark.read.csv

        spark.read.csv?
```

```
Signature: spark.read.csv(path, schema=None, sep=None, encoding=None, quote=None, escape=None, comment=None, header=None, inferSchema=
None, ignoreLeadingWhiteSpace=None, ignoreTrailingWhiteSpace=None, nullValue=None, nanValue=None, positiveInf=None, negativeInf=None,
dateFormat=None, timestampFormat=None, maxColumns=None, maxCharsPerColumn=None, maxMalformedLogPerPartition=None, mode=None,
columnNameOfCorruptRecord=None, multiLine=None, charToEscapeQuoteEscaping=None, samplingRatio=None, enforceSchema=None, emptyValue=Non
e)
Docstring:
Loads a CSV file and returns the result as a  :class:`DataFrame`.

This function will go through the input once to determine the input schema if
``inferSchema`` is enabled. To avoid going through the entire data once, disable
``inferSchema`` option or specify the schema explicitly using ``schema``.

:param path: string, or list of strings, for input path(s),
             or RDD of Strings storing CSV rows.
:param schema: an optional :class:`pyspark.sql.types.StructType` for the input schema
               or a DDL-formatted string (For example ``col0 INT, col1 DOUBLE``).
:param sep: sets a single character as a separator for each field and value.
          If None is set, it uses the default value, ``,``.
```

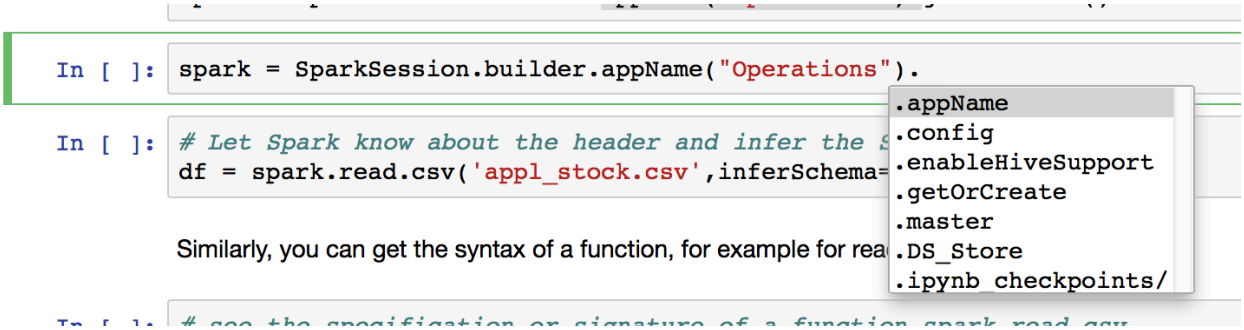
Way 2:

- Every Python object has various attributes and methods associated with it. Like with the help function discussed before, Python has a built-in dir function that returns a list of these, but the tab-completion interface is much easier to use in practice. To see a list of all available attributes of an object, you can type the name of the object followed by a period (".") character and the Tab key:

For examples, after “.” and “tab” key, you can see



And further,



- For more details about view API documents in Jupyter notebook, see

https://jakevdp.github.io/PythonDataScienceHandbook/01.01-help-and-documentation.html

Task 3: Pandas

Pandas is well suited for many different kinds of data processing or transformation, such as

- Tabular data, as in an SQL table or Excel spreadsheet
- Ordered and unordered time series data.
- Arbitrary matrix data with row and column labels
- Any other form of observational / statistical data sets. The data actually need not be labeled at all to be placed into a pandas data structure

Actions:

- Have a practice with Pandas basics, following the notebook from …Labs/Lab1/Pandas and NumPy/Pandas_basics.ipynb,

Optional:

- A tutorial of Pandas in the current directory, …/LabsLab1/Pandas_and_NumPy/tools_pandas.ipynb

1) Following this this notebook to study its basics, while running the samples.

2) A more comprehensive Pandas tutorial is given here, you may have a look at them as your brief knowledge of what Pandas does. (It takes time to learn Pandas, don't worry, we will gradually to use this tool during the course is going forward.)

3) Reference: Pandas API in Python, where you may find what you want whenever you need Pandas.

Task 4: NumPy

- **NumPy** is the fundamental package for scientific computing in Python. When we Machine Learning and Deep Learning, we need it.

Actions:

- At this stage, we just have its basics. You may follow this tutorial from the notebook, located in …Labs/Lab1/Pandas_and_NumPy/tools_numpy.ipynb, to learn understand it. (It also takes time to learn NumPy, don't worry, we will gradually to use this tool during the course is going forward.)

*Task 5: Python Basics and Exercises

**If you are very beginner of Python, you may do the following works, otherwise, you may ignore them.

1) Start *Jupyter notebook* at your home directory and then navigate to .../Labs/Lab1/Python_exercises, where you will find two documents, one for exercises (questions) and another for solutions.

2) Try to answer the questions at first. If you feel too hard, you may refer to the solutions.

3) If you are very beginner for Python, please study its basics from .../Lab1/Python_basics_notebook,

--- Edited by Chunshan Li