Tutorial #1 - Get Data0

October 23, 2018

1 Tutorial #1: Get Data

Welcome to Cognitive Class Labs Notebooks. This notebook is the first in a series of "getting started" tutorials that is designed to introduce some basic concepts and help get you familiar with using the using Notebooks in Cognitive Class Labs.

This tutorial covers:

1. An introduction to Notebooks
2. A quick tour of Cognitive Class Labs
3. Uploading files to the workbench
4. Renaming files in the workbench
5. Importing files from external URLs
6. Loading a CSV file into a `pandas` DataFrame
7. Manipulating a DataFrame

1.1 Cognitive Class Labs Notebook

Notebook is a web-based environment for interactive computing. Cognitive Class Labs Notebook is based on Jupyter (IPython) Notebooks and enables you to write and execute Python, R, Scala code within a "notebook" in your web browser. You enter code into an input cell, and when you run the cell, the notebook executes the code and prints any output to an output cell. You can change the code in an input cell and re-run the cell as often as you like. In this way, the notebook follows a Read Evaluate Print Loop paradigm.

But that's not all. The notebook also supports rendering markup cells (like this one) inline, so you can embed text, markdown, HTML, images, videos, and even interactive widgets, all within a notebook.

The flow of a notebook is top to bottom, and you can create as many cells as you desire. The interactive nature and the ability to render text and media makes Notebook a powerful environment for working with data, performing analyses, and documenting results.

Note: Our Notebooks support Python, R, Scala and other languages. In these tutorials, we assume you have a basic familiarity with the Python Programming Language

If you need more background information, we recommend these popular websites:
Learn Python the Hard Way
Using IPython for interactive work

1.2 Cognitive Class Labs Notebook

Cognitive Class Labs Notebook integrates IPython Notebook in to a cohesive ecosystem of data science tools. It extends IPython Notebook with the following capabilities:

1. Runs in the cloud, on your own servers or on your laptop
2. Allows users to tag and organize notebooks and data
3. Allows users to search notebooks
4. Enables users to share and import notebooks

1.2.1 Quick Tour

Before we get started, here is a tour of Cognitive Class Labs Notebook layout:

- **Navigation Bar**: The navigation bar at the top of the workbench page contains links to the DSWB homepage as well as dropdown menus to allow you to do everything from downloading a complete Notebook environment to your own machine, downloadinng workbench files to submiting feedback to the DSWB team.
- **Omni Box**: The box is located in the upper right, within the navigation bar. You can use it to search within notebooks in your workbook environment or to import files from publicly addressable URLs. The workbench can currently import notebooks (.ipynb) and plain text data files (.csv, .txt, ...) given a direct URL to the file. It also supports imports from NBViewer given a URL to a notebook on that site.
- **New Notebook Button**: Click the button in the top right corner to create a new notebook in your workbench. Select the language you plan to use in the notebook (e.g., Python, R, Scala).
- **Primary View**: The primary view is the center panel which renders your notebooks, displays static pages such as *Welcome* and *About*, and shows search results.
- **Sidebar**: The sidebar, located on the right side of the page, contains panels to help you organize your notebooks and data files. You can collapse or expand the sidebar by clicking the toggle (**>**) button in the top right corner of the primary view.
- **Sidebar Panels**: Sidebar panels are dynamic subpanels that are intended to help you organize your work. There are a few panels that are always shown, and of course, you can create your own custom tag panels. They can be expanded or collapsed or moved to the top/bottom of the page. The first two are default panels. You will create the others throughout these tutorials.
  - **Recent Notebooks**: Lists the most recently accessed notebooks.
  - **Recent Data**: Lists the most recently accessed data files. Currently, a data file is everything that is not a notebook (.ipynb).
  - **Recently Active Notebooks**: Lists the most recently accessed notebooks with a running IPython kernel.
  - **Custom Tag Panels**: Lists most recently accessed workbench files that have the given tag.
- **Sidebar Panel Items**: Each item within a sidebar panel represents a file in the workbench environment. Clicking on the arrow to the left of each item exposes a collapsible section below the item that contains additional details and actions you can take on the item. Hovering the mouse over any icon or link reveals more information. Clicking the arrow again hides the additional details.

- **Table of Contents**: When a notebook is open, you can click the triangle-shaped toggle button at the bottom left corner of the notebook next to scrollbar to toggle a dynamically generated table of contents. The table links to every Markdown heading within the notebook.

1.3 Where is my data?

Notebooks can work with data in variety of data sources and fomats. Every workbench comes with a local data folder where data is stored and is available for you to use within your Notebooks. This data is stored as text files using one of the popular formats (e.g. CSV, JSON etc.). In this tutorial you will bring data from external sources in to this local folder by uploading a file from your computer and importing a file from a publicly accessible repository. You can also use OpenRefine to find and prepare data for analysis and upload the files to your workbench local data folder.

In addition to the data stored in the local data folder, your code can read data from remote data sources such as relational databases and big data repositories e.g. Hadoop Distributed File System (HDFS). Access to external data is covered in other tutorials.

Very large data sets (gigabytes) are best stored and processed by big data systems such as Apache Spark, Hadoop and data warehouse systems like dashDB, DB2, Oracle, Redshift etc. Data Scientist Workbench Notebooks include Apache Spark so that you can build and test your code locally and submit it for execution to a remote Spark cluster.

1.4 Data Set

In this tutorial, and those that follow, you will get acquainted with Cognitive Class Labs Notebooks by exploring historical winter olympic medal data.

At the start of the 2010 Winter Olympics in Vancouver, The Guardian published an article to generate interest in the games by tapping into the competitive spirit of country fans. The article allowed readers to explore historical Olympic medal data from the Olympic Movement database. You can find the data **here**. We will make use of this data set in our tutorials.

1.5 Upload a File

You can easily add files to your workbench data folder by dragging them from your computer and dropping them onto the workbench.

1. Download the olympic medal data in CSV format. Click this Box link to open the document in a new browser window.
2. Save the CSV file to your computer by clicking on the Download button.
3. Drag the CSV file from your desktop onto the workbench (Note that the CSV file appears under your **Recent Data** panel in the sidebar.)

Note that the CSV file appears under your **Recent Data** panel in the sidebar.

1.6 Rename a File

The CSV file you just uploaded has a rather long name, so let's give it a shorter one. You can rename a file in the workbench data folder by following these steps:

1. Click the arrow button (**>**) next to the CSV file you just uploaded.

2. In the section that appears below the item, click "Rename"
3. Change the name of the file to "medals.csv" and press Enter or click outside the name.

1.7 Import a File

You can also import a publicly addressable file into your workbench data folder by entering its URL into the search box in the top navigation bar. The file at the URL will automatically download into your workbench data folder as long as it is:

1. accessible via HTTP or HTTPS protocol, and
2. one of the following media types:
    - Plain text
    - CSV
    - JSON (including `*.ipynb` notebooks)

1.8 Load Data

pandas is a Python package that provides data structures for managing structured data. The two primary data structures of pandas are the Series (1-dimensional) and DataFrame (2-dimensional).

In the following steps, we'll load the olympic medals by country CSV file into a DataFrame in memory.

1.8.1 Step 1: Import the `pandas` Package into our notebook.

Click on the code cell below, then click the right arrow button () in the notebook toolbar to run the code.

```
In [ ]: import pandas
```

Note: If you're a keyboard guru, you can accomplish all of the mouse actions this and other tutorials suggest via hotkeys. See the *Help -> Keyboard Shortcuts* menu item for details.

1.8.2 Step 2: Create a new code cell

Click on the plus button (**+**) in the Notebook toolbar to create a new cell.

Click the newly created cell and enter the following line of code:

1.8.3 Step 3: Insert the data file path

Place your cursor between the two quotation marks in the `read_csv('')` method call.

From the **Recent Data** panel in the sidebar, click on the arrow button (**>**) to the left of the "medals.csv" filename. Click the "**Insert Path**" link in the section that appears below the file.

1.8.4 Step 4: Run the code cell

Click on the **play** () button in the Notebook toolbar to run the code cell. The kernel will execute the code, showing a `[*]` on the left side indicating that the cell is running. Once it completes, the cell will show a number in the brackets indicating it is the Nth cell to run in the notebook.

1.9 Manipulate a DataFrame

Now that we have the data in memory, we can explore and manipulate it.

Print the first and last 5 rows of the data using the `head()` and `tail()` methods. Run each code cell below.

```
In [ ]: medals_df.head()
```

```
In [ ]: medals_df.tail()
```

The tail output shows us that the CSV file contains lines at the bottom that are not data. The cell values at these rows and columns is `NaN` (not a number).

We can prune these rows from our data by running the following code cell.

```
In [ ]: medals_df = medals_df.dropna()
        medals_df.tail()
```

Now we can sort the data by country, year, event, and type of medal. 1 sorts ascendingly and 0 sorts descendingly.

```
In [ ]: medals_df.sort_values(['NOC', 'Year', 'Event', 'Medal'], ascending=[1, 1 ,1 ,0])
```

1.10 Next

The next tutorial topic will focus on data exploration and visualization. Visit the Welcome page to download **Tutorial #2 - Explore and Visualize**.

Created by: The Cognitive Class Team