Show Submission Credentials

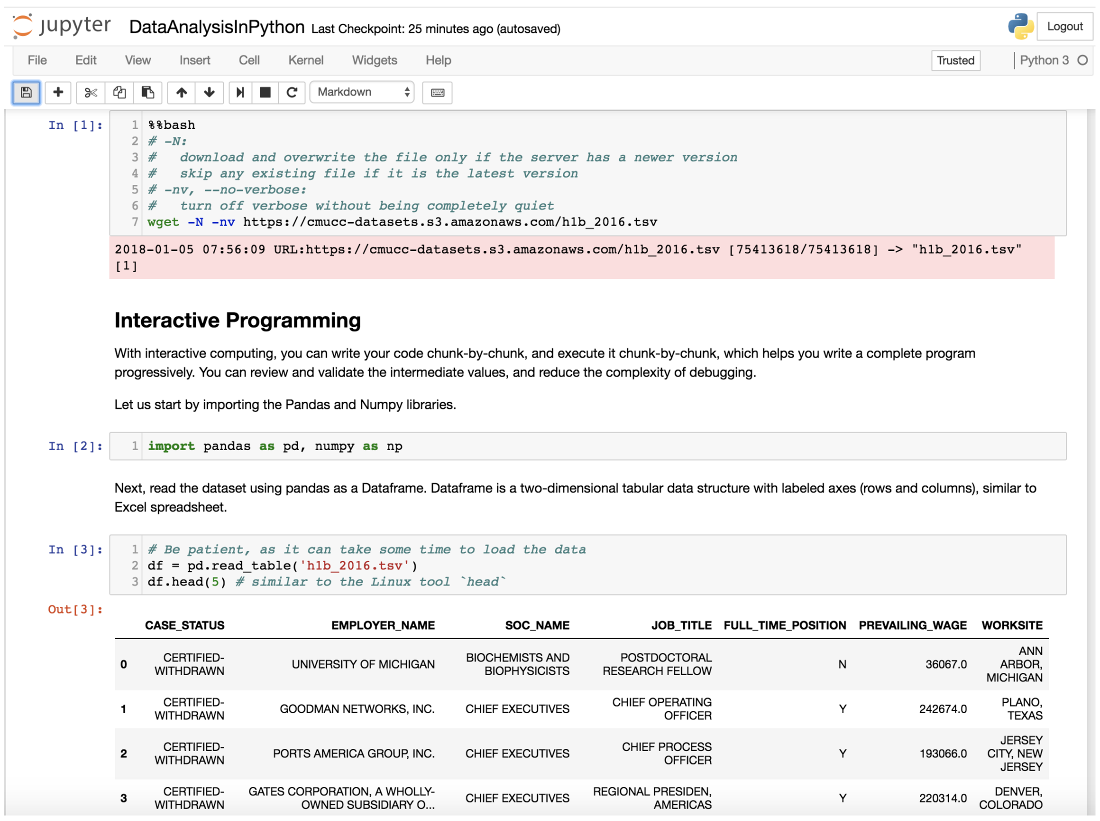
P0. Jupyter Notebook Jupyter Notebook for Data Science

Overview

Overview

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code and rich text, such as equations, links, visualizations, etc. Notebook documents are both human-readable and executable.

Did you know that Jupyter notebooks were created around three core languages? Those languages are Julia, R, and Python! While the platform is rooted in those three languages, users can explore installing any of the more than 100 "kernels" to enable your Jupyter installation to support additional languages such as Scala, Java, and Matlab among others!



In [1]:
```
%%bash
# -N:
#   download and overwrite the file only if the server has a newer version
#   skip any existing file if it is the latest version
# -nv, --no-verbose:
#   turn off verbose without being completely quiet
wget -N -nv https://cmucc-datasets.s3.amazonaws.com/h1b_2016.tsv
```

```
2018-01-05 07:56:09 URL:https://cmucc-datasets.s3.amazonaws.com/h1b_2016.tsv [75413618/75413618] -> "h1b_2016.tsv"
[1]
```

Interactive Programming

With interactive computing, you can write your code chunk-by-chunk, and execute it chunk-by-chunk, which helps you write a complete program progressively. You can review and validate the intermediate values, and reduce the complexity of debugging.

Let us start by importing the Pandas and Numpy libraries.

In [2]:
```
import pandas as pd, numpy as np
```

Next, read the dataset using pandas as a Dataframe. Dataframe is a two-dimensional tabular data structure with labeled axes (rows and columns), similar to Excel spreadsheet.

In [3]:
```
# Be patient, as it can take some time to load the data
df = pd.read_table('h1b_2016.tsv')
df.head(5) # similar to the Linux tool `head`
```

Out[3]:

	CASE_STATUS	EMPLOYER_NAME	SOC_NAME	JOB_TITLE	FULL_TIME_POSITION	PREVAILING_WAGE	WORKSITE
0	CERTIFIED-WITHDRAWN	UNIVERSITY OF MICHIGAN	BIOCHEMISTS AND BIOPHYSICISTS	POSTDOCTORAL RESEARCH FELLOW	N	36067.0	ANN ARBOR, MICHIGAN
1	CERTIFIED-WITHDRAWN	GOODMAN NETWORKS, INC.	CHIEF EXECUTIVES	CHIEF OPERATING OFFICER	Y	242674.0	PLANO, TEXAS
2	CERTIFIED-WITHDRAWN	PORTS AMERICA GROUP, INC.	CHIEF EXECUTIVES	CHIEF PROCESS OFFICER	Y	193066.0	JERSEY CITY, NEW JERSEY
3	CERTIFIED-WITHDRAWN	GATES CORPORATION, A WHOLLY-OWNED SUBSIDIARY O...	CHIEF EXECUTIVES	REGIONAL PRESIDEN, AMERICAS	Y	220314.0	DENVER, COLORADO

Why Jupyter Notebook?

Why Jupyter Notebook?

Interactive Computing

> Confront the difficult while it is still easy; accomplish the great task by a series of small acts. -- Laozi

Beginners often do not test and build their code iteratively, instead putting off debugging until their full program is implemented. When errors appear then it is often difficult to identify the source of the bug, increasing frustration and time committed to solving problems. As an example, you may be working with a very large dataset that takes 10 minutes to load into memory each time you read it from disk. Now consider if you conduct some data analysis, but you are having some bugs in that analysis code. Each time you want to debug those data analysis steps you must spend 10 minutes loading the dataset. This expensive process is difficult to work with and can be avoided by maintaining the state of your program or variables.

With Jupyter Notebook, it feels like that you can "save" your progress at the latest checkpoint, for example, you only need to load the data once because the intermediate variables are persistent and reusable. You can divide and solve the problem by its component steps. It encourages an execute-explore workflow. You can re-execute each cell as often as you like, and it makes it easy for you to explore different approaches and fix your mistakes.

Many data analysis tasks involve exploration, attempt and iteration. Jupyter will save you time and make your coding less frustrating. You can use Jupyter notebooks as a tool throughout this course to help you build code iteratively.

Persistent Output and Reproducible Analysis

As you may notice, the output is persistent after executing the code. You can write great data analysis reports to share with others. Besides, you can reproduce the analysis by re-running the code. Therefore, Jupyter excels at demonstration, research, and teaching objectives.

We have prepared several self-study, interactive notebooks for you to practice on using Jupyter notebooks! The topics cover bash scripting and data analysis in Python, which will help you to get prepared for this course.

To work on Jupyter notebooks, a Jupyter server is required.

Install and Run Jupyter Server

Install and Run Jupyter Server

Jupyter requires Python (Python 3.3+, or Python 2.7) for installation. Python 3 is strongly recommended because Python 2.7 is **no longer** maintained (EOL was in 2020).

As a server-client web application, you can edit and run your notebooks via the web browser. The application can be installed and used locally requiring no internet access or remote server.

Local Server

Download and install the latest Python 3 (https://www.python.org/downloads/), which includes `pip` ("pip installs packages") for installing Python packages.

```
# Create a virtual environment
python3 -m venv <path_to_new_virtualenv> # e.g. jupyter-env

# Activate the virtual environment
# On Unix-like systems
source <path_to_virtualenv>/bin/activate
# On Windows
<path_to_virtualenv>\Scripts\activate.bat
# To exit the virtualenv, type "deactivate"

pip3 install --upgrade pip
pip3 install jupyter

# Install other libs with pip if needed, e.g.
pip3 install pandas

# Run Jupyter notebook
# --port:   Set the port.
#           By default, the notebook server starts on port 8888.
jupyter notebook --port=8888
```

You will be prompted with the information below:

```
Copy/paste this URL into your browser when you connect for the first time,
    to login with a token:
http://localhost:8888/?token=<token>
```

You can now log in from your browser with `http://localhost:8888/?token=<token>`.

Remote Server Using SSH Tunneling

SSH tunneling creates a secure connection between a local computer and a remote machine, and it can forward a port of your local computer to a port of a remote machine so that you can visit the remote port as if you are using a local one.

We can set up a remote server on an EC2 instance.

1. Launch an instance using the latest Ubuntu Server AMI following the Amazon Web Services Intro primer and connect to the instance using SSH. Make sure port `22` is open for inbound traffic by setting the security group.

2. Connect to the ec2 instance.

   ```
   ssh -i your_pem_file.pem ubuntu@<ec2-dns>
   ```

3. Execute the commands below on the remote instance.

```
# Install pip3
sudo apt-get update
sudo apt-get install python3-pip
sudo apt-get install python3-venv

# Create a virtual environment for the user `ubuntu`
# at a directory that does not exist, e.g. jupyter-env
# Do not add `sudo` or the virtual environment will be owned by
# the `root` user instead
python3 -m venv <path_to_new_virtualenv>

# Activate the virtual environment
source <path_to_virtualenv>/bin/activate
# To exit the virtualenv, type "deactivate"

# Install jupyter
pip3 install --upgrade pip
pip3 install jupyter

# Install other libs with pip if needed, e.g.
pip3 install pandas

# Run Jupyter notebook
# --port:   Set the port.
#           By default, the notebook server starts on port 8888.
jupyter notebook --port=8888
```

4. Create a secure connection between your local computer and the Ec2 instance with SSH tunneling.

   If you are a Mac user, add the option `-L <local_port>:localhost:<remote_port>` to the `ssh` command to create the tunnel, open another terminal and run:

   ```
   ssh -i your_pem_file.pem -L 8000:localhost:8888 ubuntu@<ec2-dns>
   ```

   If you are using PuTTY on Windows, on the PuTTY configuration window, click *SSH - Tunnels* on the left pane. Enter the local port number to *Source port*, e.g. as `8000`, and set the *Destination* as `localhost:8888`. Click *Add* to add the tunneling rule to *Forwarded ports*. After the configuration, click the *Open* button to start the connection.

5. You can now log in from your browser with `http://localhost:8000/?token=<token>`.

The remote port of the remote machine that runs the Jupyter server, e.g. `8888`, **need not and should not** to be open to the public. Only port `22` is required to be open for SSH tunneling.

Google Colab

Google Colab (https://colab.research.google.com/) is a free service in that allows you to work with Jupyter networks stored on your Google Drive. There are also many public tutorials that you can "clone" as your own and run them to learn.

We have three notebooks you should complete to get prepared for this course. You can access them at the following links (must be logged in with your CMU account to view):

1. HeadFirstCommandLine.ipynb (https://colab.research.google.com/drive/1TfDWXZCHOQM6DvWHmFh1-F20BBaz5U8P?usp=sharing) is a general linux command line (bash) tutorial

2. DataAnalysisInBash.ipynb (https://colab.research.google.com/drive/11WXfJL1WR1ERZ5jszxFx_WK9p-hzQwj-?usp=sharing) is an introductory notebook in data analysis using bash builtin utilities

3. DataAnalysisinPython.ipynb (https://colab.research.google.com/drive/1lH8Nv328gxZX_1Ed0tfIm-aOK6WddspJ?usp=sharing) is a notebook introducing you to using Python and some of its available libraries for Data Analysis

Once you have opened one of these notebooks you can copy them into your own Google Drive to work with or download them as a python script or notebook file from the "File" menu. Finish these tutorials so that you will be better prepared for this course.

PyCharm Scientific Mode with Code Cells

Without a Jupyter server, you may still program in an interactive approach. Code cells are supported by PyCharm Professional Edition's scientific mode (https://blog.jetbrains.com/pycharm/2018/04/pycharm-scientific-mode-with-code-cells/) which divide a Python script into chunks that you can individually execute, maintaining the state between them.

Enjoy the interactive learning experience!