About IPython

IPython is actually what provides the interactive Python kernel we use in Jupyter notebook. And in fact, we can use IPython outside of Jupyter notebook with its command line interface in our terminal. This is very convenient and awesome for quick modifications, exploration, experimentation, and even running Python scripts!

You can read more about how IPython and Jupyter notebook work here. You can also find the official IPython documentation here.

To use IPython's command line interface, just type in `ipython` in your terminal. This should work if you already have Jupyter notebook installed. Like we always did in Jupyter notebook, let's import our packages and load in a dataset.

```
1. IPython: intro_data_analysis/census_income (python3.6)
[jlee:...data_analysis/census_income]$ ipython
Python 3.6.1 |Anaconda 4.4.0 (x86_64)| (default, May 11 2017, 13:04:09)
Type "copyright", "credits" or "license" for more information.

IPython 5.3.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import pandas as pd

In [2]: import numpy as np

In [3]: import matplotlib.pyplot as plt

In [4]: df = pd.read_csv('census_income_data.csv')
```

Even though we are in the terminal, we can still view datasets the same way with `head`.

```
1. IPython: intro_data_analysis/census_income (python3.6)

In [5]: df.head()
Out[5]:
   age          workclass  fnlwgt   education  education-num  \
0   39          State-gov   77516   Bachelors             13
1   50   Self-emp-not-inc   83311   Bachelors             13
2   38            Private  215646     HS-grad              9
3   53            Private  234721        11th              7
4   28            Private  338409   Bachelors             13

       marital-status         occupation    relationship    race      sex  \
0       Never-married       Adm-clerical   Not-in-family   White     Male
1  Married-civ-spouse    Exec-managerial         Husband   White     Male
2            Divorced  Handlers-cleaners   Not-in-family   White     Male
3  Married-civ-spouse  Handlers-cleaners         Husband   Black     Male
4  Married-civ-spouse     Prof-specialty            Wife   Black   Female

   capital-gain  capital-loss  hours-per-week  native-country  income
0          2174             0              40   United-States   <=50K
1             0             0              13   United-States   <=50K
2             0             0              40   United-States   <=50K
3             0             0              40   United-States   <=50K
4             0             0              40            Cuba   <=50K

In [6]:
```

And although you're in IPython, you can still use command line commands! So we can do things like checking our directory for other files and renaming them.

```
In [6]: ls
census_income_data.csv

In [7]: mv census_income_data.csv data.csv

In [8]: ls
data.csv

In [9]:
```

Using IPython in your terminal can be very convenient for quick changes to your files. For example, if you wanted to change the column names in a dataset before sending off a csv file.

```
In [9]: df.columns = df.columns.str.replace('-', '_')

In [10]: df.columns
Out[10]:
Index(['age', 'workclass', 'fnlwgt', 'education', 'education_num',
       'marital_status', 'occupation', 'relationship', 'race', 'sex',
       'capital_gain', 'capital_loss', 'hours_per_week', 'native_country',
       'income'],
      dtype='object')

In [11]: df.to_csv('data.csv', index=False)

In [12]:
```

Visualizations here are also the same, except there's no `% matplotlib inline` since that's specific to Jupyter notebook. To have our visualizations show, we need to call `plt.show()`.

```
1. IPython: intro_data_analysis/census_income (python3.6)
'marital-status', 'occupation', 'relationship', 'race', 'sex',
'capit
'incom
dtype='

In [8]: ls
data.csv

In [9]: df.co

In [10]: df.c
Out[10]:
Index(['age',
'marit
'capit
'incom
dtype='

In [11]: df.t

In [12]: df['
Out[12]: <mat

In [13]: plt.
```


One thing I do in IPython all the time is test or experiment with different functions, algorithms, or just Python. Sometimes, I even do a quick check here to remember how a function works before reading documentation. (Although reading documentation is a VERY good idea!)

```
In [17]: np.random.randint(1, 100, 5)
Out[17]: array([60, 84, 64, 36, 94])

In [18]:
```

This was just a quick overview to expose you to a different tool you can use to practice your new Python for data analysis skills. If you'd like to learn more, make sure to check out the documentation linked above!