Introduction to Python for Analytics

Instructor(s): David Masad

***Note**: These discussions have been selected from prior courses and lightly edited to correct typos and remove student names. A useful way to use this resource is by browsing, or via the index.*

WEEK 1

Subject: IPython vs Spyder
Participant's Question: Are we going to use Spyder in the course? Ive used it briefly once and I liked the fact that I can browse through different projects and see all my objects.
What are the pros and cons of using IPython vs Spyder and in general have you ever used Spyder in your daily job?
Instructor's Response: Great question! For anyone who isn't familiar with it -- Spyder is a Python development environment that provides a MATLAB or RStudio type of interface.
The main strength of the Jupyter (IPython) Notebooks is that they can combine text documentation, code, and output in a way that Spyder can't. For that reason, this course just uses Notebooks -- they're perfect for presenting the material, and reviewing homework submissions.
The other reason the course focuses on Jupyter Notebook is that the broader Python science and data science communities are converging around it as their tool of choice. Notebook files are easy to share, including online, and can store results and show them without needing to rerun all the code or have access to the underlying data. They also encourage good documentation, since it's easy to have text cells providing commentary explaining the code.
Finally, since Spyder uses Python's GUI libraries, it can be hard to install across different systems. The GUI is also a little more complicated than Jupyter / IPython's.
I've never used Spyder much myself, but I do think it's great for working with more complicated projects that involve multiple files. The ability to track objects that currently exist is very powerful; I've heard that a future version of Jupyter will have Spyder-type object explorer as an optional interface element.

Subject: Regarding ipython notebook
Participant's Question: Some Questions:
1. I see that every time I use Jupyter Notebook, a terminal session starts up. Is there any information in that terminal session of any importance to the programmer? It display some activity, but not all activity.
2. Also, is there a way to record a log or history of a python session in Notebooks?
Instructor's Response: Great questions!
1. By and large no, it isn't important. Mostly, the terminal tells you when new notebooks are launched, and when they save -- all information that's available in the browser notebook window itself. On rare occasions, there are errors which can crash your Python session and show notifications in the terminal. I've only ever had that happen to me a handful of times, and only using specialized libraries that call external tools -- never in pure Python, or using packages that are part of the standard scientific Python stack.
2. By default, IPython logs all commands in a session, which you can access using the ``%history`` magic (IPython-specific command). In IPython in the terminal (i.e. not via a Jupyter Notebook), pressing the up and down arrows will also scroll through your history.

Subject: FizzBuzz
Participant’s Question: Is 0 a FizzBuzz number?
Another Participant’s Response:
"When 35 is divided by 7, there is no remainder, so 35 is evenly divisibleby 7. When dividing zero by any number, the result is zero, with no remainder. Thus, by the definition of divisibility, zero is divisible by everything."
is the first thing that pops up on google, so I'm guessing yes it is
Instructor’s Response:
As mentioned above, zero has no remainder when divided by anything, so your code should FizzBuzz appropriately.

Subject: Data type of division
Participant’s Question: When I am trying to divide numbers that should come out as floats, they seem to be rounded to a whole number (e.g. 25/200=0), and I am only able to get it in the float form if I had the decimal point in (e.g. 25.0/200.0=0.125). Is there a way to set python to float the numbers? I looked a bit online, but couldn't find a solution that worked for me!
Instructor’s Response: Good question! It looks like you're using Python 2, where that's the default behavior. Ideally for this course you should be using Python 3, which knowns to convert integers to floats when doing division. The broader scientific Python community is moving towards Python 3 as the default, and all the major packages are compatible with it now.
In Python 2, you can tell it to automatically convert division results to floats with the statement:
from __future__ import division

Subject: Index of list
Participant’s Question:
In this week reading on page 7

```
In [28]: print(first_list[4:])
[12, 15]
```

I thought the result would have been [7, 12, 15] since first_list[4] = 7
Did I get this wrong?
Instructor’s Response:
Good question! Remember that the list is numbered from 0, so first_*list[4]* actually refers to the fifth element in the list, and *first_list[3] = 7.*

Subject: printing output
Participant’s Question:
I am just going through the lesson trying to reproduce it.
In my case, when I type

```
print(a,b)
# I do not get as result
a b
# I get instead
(a,b)
# also when I print("a =", 7) I do not get
a = 7
# I get instead
("a = ", 7)
```

I cannot see what I am doing wrong, also I have asked a work partner who uses programming and cannot see what is wrong.
I attach a screen shot, please look for example to In [16].

In the same line of problems, if I type
type(a/b)
# I get in response
int
# and not float, see also the screenshot In [23] and [24]
Instructor's Response:
It looks like you're running Python 2, where *print* (and division -- see my answer to Leigh, above) behave slightly differently than in Python 3. I suggest you set up Python 3 on your machine, and try running the code again.

Subject: print statement a part of the function
Participant's Question: I have a question about functions. How do I make a print statement a part of the function? I.e., I do not want to say "print my_function(arguments)". I want to simply call the function "my_function(arguments)" and have it print.
Instructor's Response:
You can put a **print** statement inside your function. For example:

```
def add(a, b):
    c = a + b
    print c
```

Will result in:

```
add(1, 3)
> 4
```

Subject: iPython notebook to PDF
Participant's Question: How do you generate a PDF file for iPython notebook?
I am able to install additional package to generate html or .latex file but what is the best way to convert this into a pdf file?
Instructor's Response: There are two ways:
The easy way (that I usually use) is to convert the notebook to HTML, then just Print to PDF. If you're comfortable working in LaTeX, you can convert to LaTeX and then compile to PDF from there. I find that that's overkill unless you really want LaTeX-quality typesetting, or to add other LaTeX features later.

Subject: python module
Participant's Question: What is "__future__" ?
Instructor's Response: Great question! It's actually a module that's part of the Python Standard Library. It's meant to document the introduction of new behaviors between Python versions, and to let you import those behaviors back into older versions.
Importing division is the main use I've ever seen it put to. But a fun thing to try is:
from __future__ import braces

Subject: Python Packages

Participant's Question: I am able to setup iPython environment using the anaconda package. But have few basic questions about Python package installation.
My question confirmation and finding what is the best way to manage Python packages:
1. I found there are number of ways to install Python packages
    a. Using pip
    b. Using easy_install but many website are recommending against this as it may not be secured.
    c. directly downloading tar file and running setup.py file
2. Also is it possible to run iPython environment on a remote machine and provide access to multiple users remotely through a browser?
Instructor's Response: Both good questions!
To answer your first question: You're right, *easy_install* isn't the recommended way to go anymore. Usually, there's no difference in the results between using *pip* and installing from source, but pip is a bit user-friendlier and can install dependencies automatically for you. Another advantage of pip is that it's meant to work with another Python feature, Virtual Environments, that lets you have multiple versions of packages installed and switch between them. We won't get to it in this course, since it's mostly useful for web developers who need different Python setups that are compatible with different server deployments. If you're using virtual environments, pip knows to install packages for a specific environment only.
Bottom line: I suggest using pip first, downloading from source if pip doesn't work, and only using easy_install if the instructions specifically tell you to.
Instructor Continued: To answer your second question:
Yes, it is possible to install IPython on a remote machine and access it through a browser. However, that feature isn't fully developed yet, so IPython can't really do collaborate editing or even track which user is using what file.
Basically, what you would do is just run the IPython Notebook on your remote server, then (on the server-side) open port 8888 (by default) to remote connections. Then users can access it by pointing their browsers at port 8888 on the remote server's address.
For security purposes, you should put a password on your notebook: http://ipython.org/ipython-doc/rel-0.13.1/interactive/htmlnotebook.html#security
Here's an online discussion thread that goes into some more details:
http://www.reddit.com/r/IPython/comments/1dff7w/question_on_making_an_ipython_notebook_server/
Bottom line: it's possible, but unfortunately this feature could still be improved.

Week 2

Subject: Derived/Calculated columns for 'read in' file
Participant's Question:
Is it possible to create derived/calculated columns when a file is read in? For example, if I have a date column when I read it in I want to create three extra columns - day, month and year by passing it through a function when reading in.
Instructor's Response:
Good question! There's no simple built-in way to do that, but you could write your code to, for example, compute additional values for each row as it's read it and append them to that row's list. With pandas, which we'll cover next week, you can easily add columns with values based on other columns once the data is read in.

Subject: Dictionaries

Participant's Question:
Assuming I have a csv file with seven(7) columns with data on employees of a company. The columns are namely, ID, First Name, Last Name, Age, Salary, Nationality and Marital Status. I understand how to open this csv file and read it into a file. Hence the file will have data from the seven columns. I also understand how to create a list with only one of these column. What I do not understand is if I want to create a list with just data from the 'Last Name" column and the "Salary" column, How do I go about doing this?
Instructor's Response: One thing you can do is use list comprehension to create a new list of lists, with only the two columns you care about. For example:
subset = [[row[0], row[5]] for row in data]
Alternatively, you can create two column lists, and group them into a list:

```
col_1 = [row[0] for row in data]
col_2 = [row[5] for row in data]
subset = [col_1, col_2]
```

However, in Week 3 we go into pandas, which provides much more user-friendly ways of selecting columns.

Subject: structure of an object
Participant's Question:
How do you determine the structure of an object?
Instructor's Response: One way is to just have Python show you the object, by typing it on its own line and hitting enter. A list will have [square, brackets], a dictionary will have {curly: brackets}.
Alternatively, you can use the *type(...)* function, which will look something like:

```
 >>> type([1, 2])
<type 'list'>
 >>> type({"a": 1})
 <type 'dict'>
```

Subject: built in modules and functions
Participant's Question: Where should we go to find out about built in modules and functions, please?
Instructor's Response: The official documentation for the Python Standard Library is at: http://docs.python.org/2/library/

Subject: reading semi-structured data
Participant's Question: Are there some tips on reading semi-structured data such as json, twitter, rss feeds, web pages or text information? How do you read such data in python and also do exception handling from local files or directly from online resource would be helpful?
Instructor's Response: Different file formats require very different tools and methods.
Python has a JSON parser as part of its standard library (http://docs.python.org/2/library/json.html) which can read and write between JSON files and Python objects like dicts and lists.
Web scraping is a big topic -- it could probably be its own course! The two most important libraries are probably Requests (http://docs.python-requests.org/en/latest/) for loading pages, and BeautifulSoup (http://www.crummy.com/software/BeautifulSoup/) for parsing them.
There is a Python wrapper around the Twitter API; the data itself comes back as JSON, so you just need to parse that. I've never worked with RSS, but it looks like there are several Python libraries for that as well.

Week 3

Subject: getting help
Participant's Question: Hi, what's the most direct way to get help on a function or package in Python? In R you can type "?apply" on the command line and it'll open an HTML "dictionary". Thanks.
Instructor's Response: In Jupyter Notebooks, you can put the question mark after a command or function and then run to have the documentation pop up at the bottom. e.g. *list?*
In all Python environments, you can use the built-in *help* function, e.g. *help(len)*
Note that what it's giving you is called the docstring -- this is actually the documentation put at the top of a function or class, including ones that you've written yourself. For example, suppose you write your own function:

```
def my_add(a, b):
    ''' Add a and b together.
    Args:
        a, b: Two numbers
    Returns:
       The sum of a and b
    '''
    return a + b
```

If you forgot what your function does (or if someone else wants to use it without reading through all of your source code), the command *help(my_add)* or in Jupyter *my_add?* will bring up the text in the triple quotes.

Subject: single versus double quote marks
Participant's Question: In Python, is there a difference between a single and a double quote mark for string literals?
Instructor's Response: Not really. The only difference is that if you want to include singe-quotes in a string, you'd use double-quotes to enclose it, and vice-versa. For example:
some_string = 'We can include "double-quotation marks" in this string.'
other_string = "And in this one we can only have 'single' quotes."
The one type of quote that is actually different is triple single- or double-quotes. These allow you to have multi-line strings, for example:
long_string = '''I have eaten
the plums
that were in
the icebox'''
This is exactly the same as:
long_string = """I have eaten
the plums
that were in
the icebox"""

Subject: looping through a DataFrame
Participant's Question: In the section "Subsetting" we're fixing the negative values inserted previously in the movies["Non_US_Gross"] column. To loop through this pandas dataframe and avoid creating these negatives originally created, I tried running the following code but it didn't work:

```
for i in movies:
    if movies["Worldwide Gross"][i] == 0:
        movies["Non_US_Gross"] = 0
    else:
        movies["Non_US_Gross"] = movies["Worldwide Gross"] - movies["US Gross"]
```

What am I doing wrong? And, is there a better way to do this? I found in cell "In [81]" an example of how to loop through a DataFrame but I couldn't really get it. Any guidance would be appreciated.

Instructor's Response: There are a few things going on here. When you for-loop over a dataframe, you're actually looping over column names. So with *for i in movies*, i takes on the values "Release_Date", "Movie", etc. rather than the index values.
If you want to loop over index values, you need to do it as *for i in movies.index:*
Another issue with your code is that, after the second line, you leave the index off. Operations on a column without an index specified operate on the entire column. So in your code,

```
    if movies["Worldwide Gross"][i] == 0:
        movies["Non_US_Gross"] = 0
```

actually ends up meaning that if any row has a Worldwide Gross of 0, the entire Non_US_Gross column is set to 0. Similarly,

```
 movies["Non_US_Gross"] = movies["Worldwide Gross"] - movies["US Gross"]
```

Also sets the entire Non_US_Gross column equal to "Worldwide Gross" - "US Gross", for the values in each row.
More broadly, though, the suggested way of working with pandas Series and DataFrame columns is via full-column operations rather than loops. (This is similar to vectors in R, Julia, or other vectorized languages). See Cell 75 in this week's lesson for an example of how to do this.

Subject:
Participant's Question: In cell "In [63]" from the lecture, we have the code: movies = pandas.read_csv("MovieData.csv", sep='\t', na_values = ["Unknown","Unkno"], parse_dates=[0]), when asking pandas to parse the column as dates. Does the zero value argument in parse_dates refer to the column position in the dataframe? Or does this zero value have a different meaning?
Instructor's Response: You got it -- the zero value refers to the column position.

Subject: Regarding Ipython notebook
Participant's Question: My question is more about jupyter / iPython than language particulars. How do we change the working directory while working in a jupyter notebook? I suppose we could use the magic command '%cd' But you first must be in a notebook to do so. Also, I have picked up on Stackoverflow that we can change some configuration file, so jupyter opens in the same working directory. Do you know anything about that?
Instructor's Response: By default, the working directory for each specific Notebook is the directory it was created in. When you launch Jupyter, you should be able to navigate through your directory tree in the interface itself, and create a new notebook file in whatever directory you've currently opened.

If you're launching Jupyter Notebook from the command line, the starting directory is wherever you launched it from. You can put your new notebook files anywhere there or below, but not above. If you're using Anaconda's launcher, you do need to edit a configuration file. Take a look at this StackOverflow answer.

Subject: what is returned from *.info()
Participant's Question: So, I am coming from a R, SAS Java.. background, and there are a few things about Python that I am still getting use to. And the latest mystery to me is that there are certain functions whose output cannot be saved.
For example, consider the dataframe, say df, the one we make in this week's lecture notes. The output of df.info() cannot be saved to another object, at least from what I see. If we have the statement x=df.info(), all of the output is displayed on the interactive screen, but nothing is stored in x.
So, (a) why not? and (b) I may want to used the stuff returned from df.info() and use it in some batch/non-interactive program. I guess I am so use to R where everything is either an object or a function. And when we simple type in the name of an object, a print() function belonging to the class of the object is dispatched and executed. What is happening in Python? Why is not x full of what we see on the screen?
Instructor's Response: That's a good question! I had never actually realized that the .info() method isn't assignable. In general, that's unusual for pandas -- have you come across anything else that behaves similarly?
The general answer is that functions need to return something for it to be assigned to a variable. If a function or method doesn't return anything, trying to assign it will result in a None value. For example:

```
def eg_function(a):
    print(a)
v = eg_function(1)
# Output will be 1; v == None
On the other hand:


def eg_function(a):
    print(a)
    return a
v = eg_function(1)
# Output will be 1; v == 1
```

So the simple answer to your question (a) is that DataFrame.info() doesn't actually return any value. I can't tell you why the pandas designers made that decision, though.
As to (b) -- you can still acquire and store the same information as .info() outputs separately. For example, df.dtypes returns the type of each column, and df.count() returns the number of non-null elements in each column. You can assign these to variables and work with them.
For some reason, .info() also allows you to write the output to a file instead of printing it. If you need to store the output of that specifically, you can do:

```
with open("df_info.txt", "w") as f:
    df.info(buf=f)
```

And that will write the output to the df_info.txt file.

Subject: Decade considerations
Participant's Question: I check the definition from wiki and got the information below (https://en.wikipedia.org/wiki/Decade#Distinctions).

"Although any period of 10 years is a decade,[1][2] a convenient and frequently referenced interval is based on the tens digit of a calendar year, as in using "1960s" to represent the decade from 1960 to 1969.[3][4] Often, for brevity, only the tens part is mentioned (*60s* or *sixties*), although this may leave it uncertain which century is meant. These references are frequently used to encapsulate popular culture or other widespread phenomena that dominated such a decade, as in *The Great Depression of the 1930s*.
Because the common calendar starts with year 1, its *first* full decade is the years 1 to 10, the *second* decade from 11 to 20, and so on.[5] So although the "1960s" comprises the years 1960 to 1969, the "197th decade" spans 1961 to 1970."
Now, I feel confused about 1960s and 1970s. Can someone please comment on it?
personally, I prefer 1960 to 1969, 1970 to 1979, 1980 to 1989, etc.
Instructor's Response:
I've never actually seen anyone actually using the 1961-1970 definition (even if it might technically be more correct). For this assignment, use the commonly accepted definition: the 60s are 60-69, 70s are 70-79, etc.

Subject: Format Character "%r"
Participant's Question: I am confused about '%r' format character. For what type of values it is used ? It is not applying '\n' character in my code. Please help.
Instructor's Response:
In general, "%r" is a way of converting any object into a string, for inserting into another string. With most built-in objects, the conversion is obvious -- for example, converting the number 2 into the character "2". It doesn't automatically insert the end-of-line character.
Can you give a code sample, or some more information on what exactly you're trying to do? That might help me give you more specific help.
Python 3 introduced a new way of formatting strings, which you may also want to take a look at. You can read about in the Python documentation, here: https://docs.python.org/3.5/library/string.html#format-string-syntax
Participant Continued: Thank you for your reply.
My code is as:

```
training1 = "Monday\nTuesday\nWednesday"
training2 = "Thursday\nFriday\nSaturday"

print training1
```

I got Output as below:

Monday
Tuesday
Wednesday

Then I tried the following code:

```
print "For which post you are applying? \n Planning Projects
OR Marketting"
post = raw_input()
if "plan" in post.lower():
    print "Your training will be on following days %r"
%training1
```

```
if "market" in post.lower():
    print "Your training will be on following days %r"
%training2
```

Output below:

For what post you are applying?
Planning Projects OR Marketing

When I typed: planning projects

Output below:
Your training will be on following days 'Monday\nTuesday\nWednesday'

'\n' work previously in print command but didn't work with format character.
Instructor's Response:
You've found an interesting edge case. Basically, every Python object can be converted to a string in two ways: str (string, which is "%s" as a format character) and repr (representation, "%r"). The purpose of str is to be readable, and str is supposed to be unambiguous. Most of the time, these end up being the same thing. But in some cases -- such as strings with special characters, like you have here -- there's a difference. In your case, the repr includes the special characters explicitly, while the str would parse them appropriately.
So if you replace "%r" with "%s" in your print commands, it should work the way you want it to.

Subject: Dictionaries
Participant's Question: When you have a time series, the autocorrelation is 1 for t=0, and various values for t = 1, 2, 3,
Which value does the ts1.autocorrelation represent, please?
Instructor's Response:
.autocorr() defaults to lag-1 autocorrelation.

To get different lags, you need to use the *.shift(..)* method, which shifts a series a specified number of units left or right. For example, lag-4 autocorrelation could be computed like this:

```
> ts.corr(ts.shift(-4))
```

Subject: Autocorrelation
Participant's Question: Assuming I have a csv file with seven(7) columns with data on employees of a company. The columns are namely, ID, First Name, Last Name, Age, Salary, Nationality and Marital Status. I understand how to open this csv file and read it into a file. Hence the file will have data from the seven columns. I also understand how to create a list with only one of these column. What I do not understand is if I want to create a list with just data from the 'Last Name" column and the "Salary" column, How do I go about doing this?
Instructor's Response:

Subject: ipython notebook

Participant's Question: 1. I think the ipython notebook idea is great, is there any way to switch directories once its open and/or open from the MAC gui rather than in terminal?
2. I notice that sometimes a comment is initiated/ended with ''' rather than #. Is the former just to enable multi-line comments?
3. I also noticed that sometimes the argument to a function has an "r" in it, e.g., pattern = re.compile(r'';|, '')
Instructor's Response:
1. Nope, right now you can only change directories based on where you launch the terminal, since IPython is actually running a local web server from that directory. You can import other notebooks into that directory by dragging them into the Notebook list page.
2. Yes, you can enclose multi-line comments inside triple single or double quotation marks. When those comments come at the beginning of a function or a class, they are called a docstring, and can be viewed from the interpreter to provide documentation on how to use that function or object.
3)'r' only comes in the context of regular expressions; it's just a way of telling the interpreter to expect regular expression code.

Week 4:

Subject: about matplotlib
Participant's Question: Last week we also learnt about using the help() command to access Python objects documentation. Sometimes the documentation can be too much to handle. So, my questions are:
1- How would I know the other types of plots in matplotlib? Where would I find this?
2- Also, these great libraries we've been learning about, how often may they change, in your experience? When they do, do they mostly add more features or change existing ones? My fear is that if some substantial work is developed using any of these libraries then it could become costly to maintain in the long run.
Instructor's Response: 1. A great starting point is the official matplotlib gallery. It has a variety of examples, along with the code snippets used to create them. More broadly, most packages have fairly good online documentation. Their websites are always a good place to start looking for documentation, examples, etc.
2. The libraries we've covered in this course are fairly solid, and are unlikely to break backwards-compatibility. Matplotlib, for example, is planning a big version 2 release -- and the big change that most people will notice will be a change of the default color scheme. pandas is a bit more likely to break compatibility, since it's technically before a 1.0 release (which is traditionally when backwards compatibility is preserved from). However every new release documents fairly carefully what changes have been made: for example, here's the most recent one.
Even as libraries are updated, you always have the option of installing a particular older version. This is where a more advanced Python feature called virtual environments becomes especially important. Virtual environments allow you to maintain a certain set of Python libraries, so that you can have different environments each with different combinations of libraries, and different versions of the same library. If someone else wants to use a project developed with older libraries, they can set up their own virtual environment and install older versions of those libraries into it, without affecting the rest of their Python setup.

Subject: apply() and lambda
Participant's Question: I have another question that is an issue I am just curious about--though the question is closely related to the assignment. I was trying to convert my Release_Date time stamps to a string by using the apply method and a lambda function as follows:

```
movie['dateString'] = movie.Release_Date.apply[lambda sd: sd.strftime('%Y-%m-%d')]
```

I received the following error:

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-74-e386fe20dbb8> in <module>()
----> 1 movie['dateString'] = movie.Release_Date.apply[lambda sd: sd.strftime('%Y-%m-
%d')]
TypeError: 'method' object is not subscriptable
```

I followed this with the following test that works fine:

```
In [75]: testDate = movie.Release_Date[1].strftime('%Y-%m-%d')
In [76]: testDate
Out[76]: '2007-05-25'
```

My question is: Why does the apply method fail ('object is not subscriptable') while the test seems to suggest to me that this should work?

Instructor's Response: It looks like you're using square brackets on .apply instead of regular parens. So instead of .apply[...] it should be .apply(...)
'Object not subscriptable' means that you're trying to use square brackets somewhere they don't belong.

Subject: Other resources
Participant's Question: as we enter the data science world, equipped with the knowledge we learned from this course, what do you consider to be some of the best resources that we can use to further our skills at Python with a focus on data science that could best serve us on the way..
I subscribe to Safari books online and I can pull up maybe 300 books recently written on Python and specialized applications within Python that they cover, not including the additional books I stumble upon at Amazon.
No one can read them all. What books and other resources do you find most helpful?
Instructor's Response: Beyond the basics, "data science" is such a huge field that it's hard to recommend a best resource. A lot depends on what problems you're trying to solve.
Though I can't recommend specific books, I can point you towards some tools that I think are useful. I suggest starting with their documentation, and looking for books that address or utilize them.
Scikt-Learn (you'll see it called SKLearn sometimes for short) is the primary machine learning toolkit for Python. It's mature and well-documented, and implements a wide variety of machine learning algorithms, as well as tools for things like data preprocessing, cross-validation, and hyperparameter search. Some of the deep learning toolkits (Google's TensorFlow in particular) offer APIs that are meant to be compatible with Scikit-Learn's.
statsmodels is a library for more traditional econometrics and statistics. The documentation and user base aren't quite as robust as SKLearn's, but it has some powerful features. In

particular, it's better than SKLearn for cases where you want to estimate and interpret regression coefficients -- and especially if you want to output a familiar regression table (see this example)
PyMC (and PyMC3, the development version) implement Bayesian statistics and probabilistic programming in Python. There's an excellent free ebook, Bayesian Methods for Hackers, that goes into both the stats and programming sides of it.
Outside of Python proper, there are a few important skills that are generally important for getting data science done. One of these is just using the Unix / Linux command line, and shell scripting (this is an area I always have to look things up for). Most data science computers and clusters will be running Linux, especially if you're using AWS or another cloud service. So often, the first step of a Big Data type project will be setting up a new cloud instance or cluster, which generally requires using the command line.
Another important skill is version control -- especially Git and GitHub. If you haven't encountered it -- version control is like Word's Track Changes on steroids. It lets you save every version of your project along the way, roll back easily to any previous version, or split a project into multiple 'branches' you can modify independently, then merge them back together. It also does this across multiple users, so that multiple people can work on the same project and combine their work without emailing files back and forth. Version control makes it easier to tinker with your own code, and is indispensable for any kind of collaborative coding. So many data science projects are open-source and hosted on GitHub, so knowing how to work with them can be helpful too.
Hope that helps! If you have a specific area or type of project you're interested in, I can try to help point you towards more specific relevant tools and resources.

Subject: datetime and %%time
Participant's Question: What implies %%time? Are "%%time" and "import datetime" same? Do they perform same operations?
Instructor's Response: datetime is just the Python standard library for handling date and time objects. Commands starting with '%' or '%%' are IPython "magics," not part of the Python language itself. You can put %%time at the top of an IPython Notebook cell to find out how long the code in that cell takes to execute.

Subject: Data Analysis in python
Participant's Question: Suppose I have a csv file with 100 rows. The columns are sex, age, salary, marital_status and education_level. I can perform frequency table and cross tabulation of these variables in SPSS and R, however, I will like to perform these same operations in python. For instance, what if I want
1) a frequency table for marital_status
2) a cross tabulation of sex and educational level
3) create a new variable "age group category" and place the age of each row in this new column e.g age = 20 would go into age group category = "18 - 25"
Instructor's Response: For a frequency table, you use the *Series.value_counts()* method, for example:
marital_status.value_counts()
For cross tabulations, you use pandas.crosstab(...), e.g.:
pandas.crosstab(df.sex, df.education)
For categories, there are a couple things you can do. If the categories are numeric, you can use the *pandas.cut(...)* method, for example:

```
s = pd.Series(np.random.randint(0, 100, 100)) # Create 100 random integers
between 0 and 10
categories = [0, 18, 25, 39, 65, 100] # Cut points for categories
binned_data = pd.cut(s, categories)
# Then binned_data looks like:
0       (39, 65]
1       (25, 39]
2       (39, 65]
3       (39, 65]
4      (65, 100]
5      (65, 100]
```

If the data isn't numeric, the easiest way is just to select the rows that you want, and assign the new variable value to them.

Subject: Trouble with plotting legend
Participant's Question: I am confused about how the legend works, particularly with a secondary axis. I tried two alternatives, one with two separate time series and one with a pandas dataframe but I got stuck in both.

OPTION 1

```
fig = plt.figure(figsize=(8,6))
ax = fig.add_subplot(111)
ax.set_ylabel("Number of Movies")
monthly_count_series.plot(ax=ax, c='b', linewidth=2)
ax2 = monthly_profitability_series.plot(ax=ax, secondary_y=True, c='k') # Create the
secondary y axis
ax2.set_ylabel("Average Profitability")
ax.legend(["Number of Movies"], loc='upper left')
ax2.legend(["Average Profitability"], loc=upper left')
```

Not very surprisingly, this gives me only ax2.legend (I can use a different position to separate ax and ax2 legends but then the figure looks very odd with the legend in two places)

OPTION 2

```
fig = plt.figure(figsize=(8,6))
ax = fig.add_subplot(111)
monthly_profitability.plot(ax=ax, secondary_y=["Average Profitability"], color=['b', 'k'])
ax.set_ylabel("Number of Movies")
```

This gives me a legend but by default it puts it on the upper right. I want it in a different location, and using
ax.legend(["Number of Movies", "Average Profitability"], loc='upper left')
gives me only the number of movies. I want a line for the average profitability as well.

Instructor's Response: You've run across a tricky issue -- secondary axes can be a pain to deal with. There are a few ways of creating the legend from both axes, but all of them boil down to getting the objects representing the lines and labels from both axes, and combining them into a new legend.

Here's one way to do it:

```
df = pandas.DataFrame({"Monthly_Profit": monthly_profit,
                       "Monthly_Count": monthly_count})
fig = plt.figure(figsize=(8,6))
ax = fig.add_subplot(111)
axes = df.plot(ax=ax, secondary_y="Monthly_Profit")
h1, l1 = axes.get_legend_handles_labels()
h2, l2 = axes.right_ax.get_legend_handles_labels()
# Note that both the h and l objects are lists:
lines = h1 + h2
labels = l2 + l2
ax.legend(lines, labels, loc='lower right')
```

Subject: missing values
Participant's Question: In regard to missing values; do these always have to be filled in, or is there a way to tell Python to analyse the data, while essentially ignoring these, or at least remove those points from the dataset and flag them as 'missing'?
Instructor's Response: Good question about missing values. There are a couple ways of dealing with them. As you mention, one is replacing them; another is selecting only the rows/records that have non-missing values, for example, *my_series[my_series.notnull()].mean()*. There are also a few numpy functions which explicitly ignore NaNs (Not-a-Number, which is how missing numeric values are usually stored), like nanmean and nansum. You may also want to check out the section on Working With Missing Data in the pandas documentation.
For histograms, I think you generally want either the labels at the bin breaks the code gives you, or else at the bins you set yourself, e.g. *plt.hist(my_data, bins=[0, 5, 6, 7, 8, 10])*
For bar charts more generally, you need to fiddle with the *xticks* (or *yticks*, if you're labeling the y axis). Here's a simple example which demonstrates how to do it: http://matplotlib.org/examples/api/barchart_demo.html

INDEX