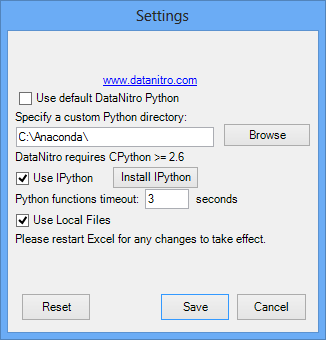
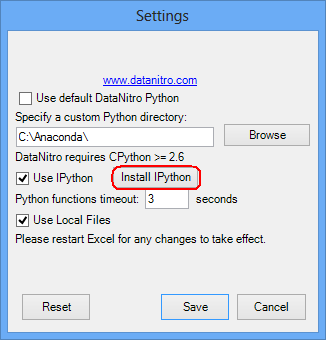
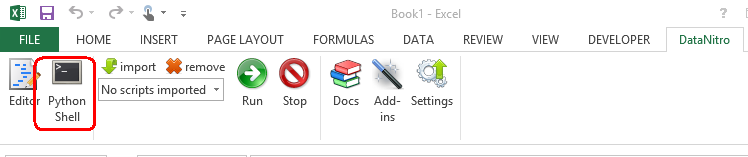
Installation & Setup

1. Install PowerQuery for Excel 2013 and the Anaconda Python distribution.
2. Install matplotlib version 1.3.0 from the matplotlib site: <http://matplotlib.org/downloads.html>  
   Use the version ending in amd64-py2.7.exe if you installed 64-bit Python, and the version ending in win32-py2.7.exe if you installed 32-bit Python.
3. Download and install DataNitro. Log in at <https://datanitro.com/pro/auth/login> with  
   We have free home licenses for students and academics. You can get one by [registering with your school email address](https://datanitro.com/pro/register), and then contacting us at [support@datanitro.com](mailto:support@datanitro.com).   
    For the trainers, please contact Wenming.  
     
    Install for the current user, and don’t install Python.
4. Open Excel, press the settings button in the DataNitro tab, and change the Python path to your Anaconda folder. Restart Excel.
5. Press “Install IPython” in the settings menu.

You should be all set!

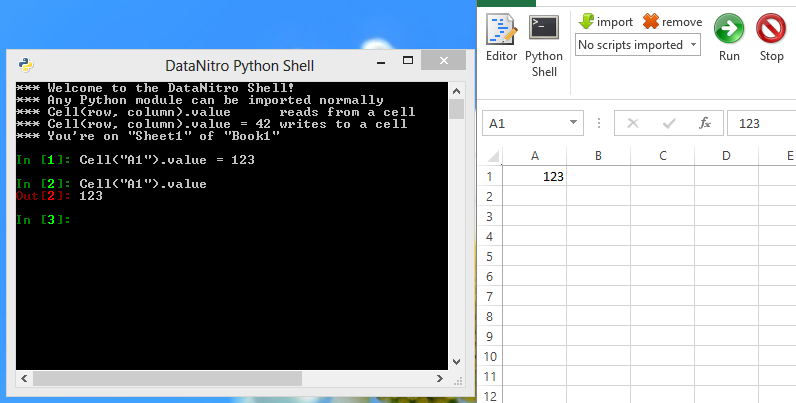
Presentation

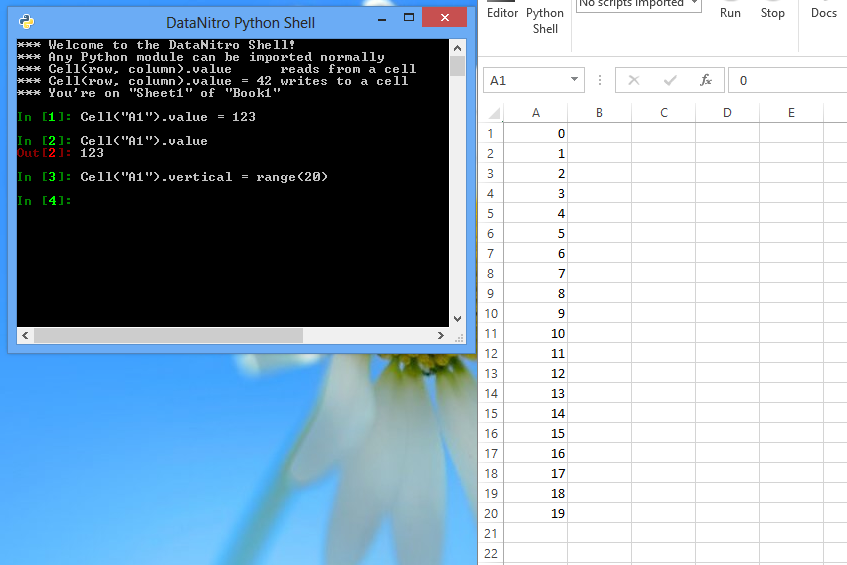
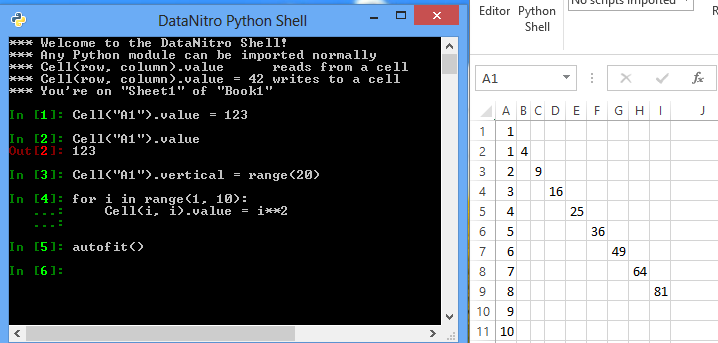
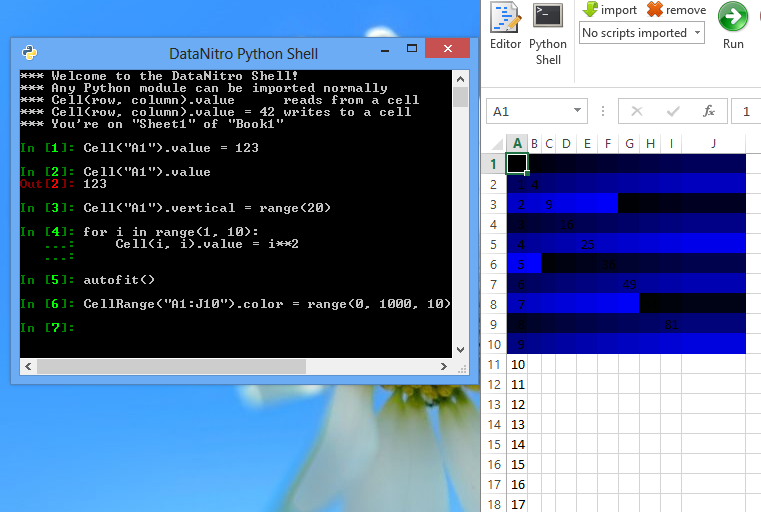
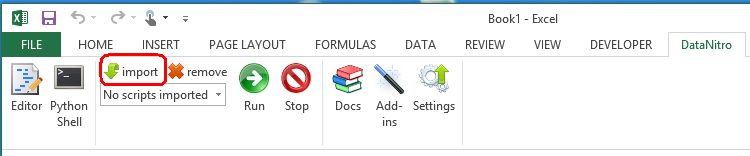
1. It helps to increase the zoom level in Excel, and the font size in the Python shell and editor.  
   You can increase the shell font size by opening the shell, right clicking the shell window, pressing properties, and going to the font tab.  
     
   You can increase the editor font size by pressing editor, and going to options.  
     
   Open a blank workbook for this demo.
2. What is DataNitro?  
   **Goal:** Explain DataNitro, why it’s useful, and how it works.  
     
   **Talking points:**
   * Python has a cutting-edge scientific computing stack, including Numpy and Scipy as an alternative to MatLab, and Pandas as an alternative to R.  
     It also offers unparalleled development speed, including speeding up data cleaning from multiple sources.  
     (This is a good place to go over Python talking points from the previous day, as appropriate.)
   * Excel is a great tool for sharing and displaying data, and letting less-technical colleagues interact with and explore data on their own.
   * DataNitro brings these two worlds together. The goal is to provide a dead-simple interface between Excel and Python.

**Demo:** Basic DataNitro Syntax

Open the Python shell, and type the following commands, one at a time. Tell the audience what’s happening as you type.

The goal is to highlight the simplicity of the API.

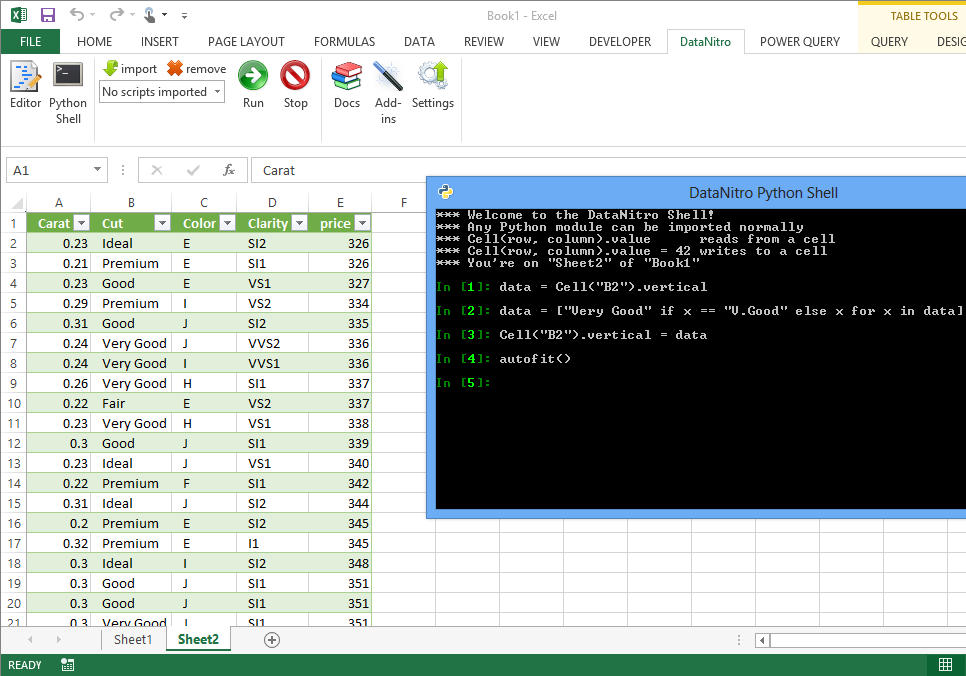
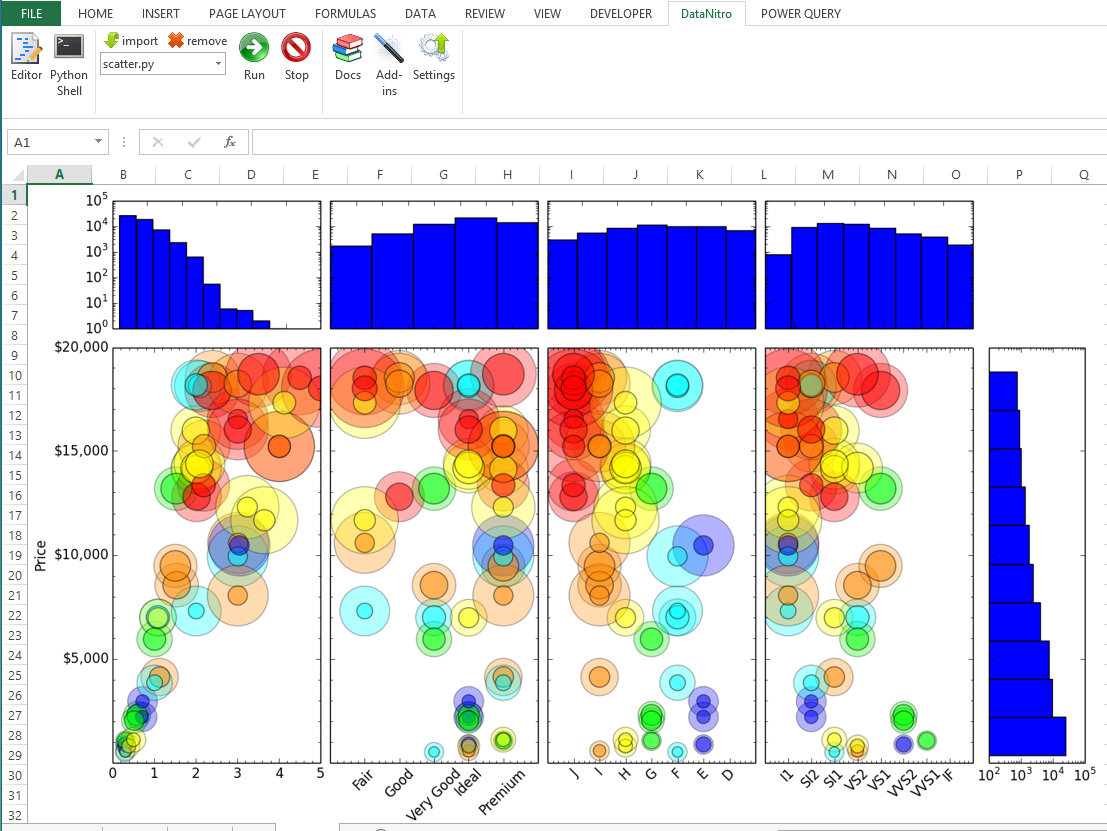
1. Basic reading and writing:  
   In [1]: Cell(“A1”).value = 123  
   In [2]: Cell(“A1”).value  
   Out[2]: 123

1. Writing a column  
   In [3]: Cell(“A1”).vertical = range(20)  
   
2. Writing a diagonal  
   In [4]: for i in range(1, 10):  
    …: Cell(i, i).value = i\*\*2  
    …:  
   In [5]: autofit()  
     
     
     
     
     
     
     
     
     
     
     
     
   Key point: Programming syntax is more versatile than using a GUI when it comes to complex actions.
3. Changing colors  
     
   In [6]: CellRange(“A1:J10”).color = range(0, 1000, 10)  
   Key points:   
   You can edit any property of a cell with DataNitro.   
     
   
4. Close the python shell.
5. Importing scripts  
     
   **Talking points:**  
   In addition to the shell, you can write and run DataNitro scripts. We’ll be using 3 scripts to analyze some data.  
     
   **Instructions:** Press “import” in the DataNitro shell, and import the scripts “scatter.py”, “predict.py”, and “sk.py” from the “diamond demo” folder.
6. Importing Data  
   **Goal:** Download data from Azure.

**Instructions:**  
Press “From Web” in Power Query, and type in: [http://datanitro1.cloudapp.net/cut\_diamonds.csv](http://www.google.com/url?q=http%3A%2F%2Fdatanitro1.cloudapp.net%2Fcut_diamonds.csv&sa=D&sntz=1&usg=AFQjCNEfU7aMnfUjVbcoU_3LEWvSeRNgfg)

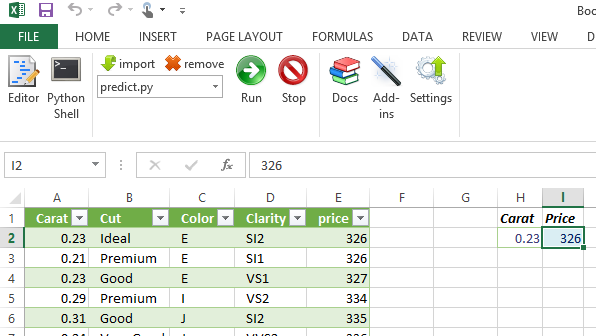
Press “Done” to import the data directly.

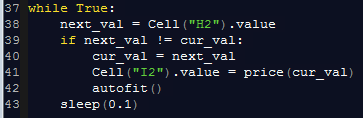
**Talking points:**  
This is our raw data, stored on an azure instance. We’re using Power Query to import it into excel.  
We’re looking at a list of diamond prices. Diamonds have four major characteristics:  
 *Carat* (how large they are)  
*Cut* (how well they’re cut from raw stones) *Color* (how clear they are)  
*Clarity* (lack of internal blemishes)  
  
We’ll use Python to understand how these traits affect a diamond’s price.

1. Data Cleaning  
   **Goal:** Demonstrate data cleaning with Python  
   **Instructions:** With the sheet containing the diamond data visible, open a new Python shell, and type the following:  
      
   In[1]: data = Cell(“B2”).vertical  
   In[2]: data = [“Very Good” if x == “V.Good” else x for x in data]  
   In[3]: Cell(“B2”).vertical = data  
   In[4]: autofit()  
     
   **Talking Points:**  
   One of the values, “V.Good”, is shortened from its full name, “Very Good”. We want to use the full name, so we’re going to use a list comprehension to read in the data, transform it, and write it back.  
     
   (close the Python shell)
2. Graphing  
   **Goal:** Showcase matplotlib in Excel.  
     
   **Instructions:**  
   Select “scatter.py” in the datanitro toolbar’s dropdown, and press “Run”.  
     
   The following chart should appear:  
     
     
    **Talking Points:**   
   Graphing is one of the best ways to understand a complicate dataset. We’re using matplotlib to make something more sophisticated than Excel’s native charts.  
     
   There are two types of graphs here. The first are histograms showing the distribution of diamonds along each dimension (carat, cut, color, clarity, and price, respectively). Each histogram is plotted on a log-scale. This shows us that there’s an exponential distribution of diamonds by price, and a super-exponential distribution by carats.   
     
   The second are scatter plots of each independent variable graphed against price. Each point shows the carat weight, color, and clarity of the diamond it represents:
   * The radius of the point is based on the carat weight;
   * The color of the point is based on the color of the diamond (with higher frequency colors representing better coloring – e.g. violet is better than red)
   * The darker portion of each point is based on the clarity of the stone, with a bigger inner circle representing clearer stones

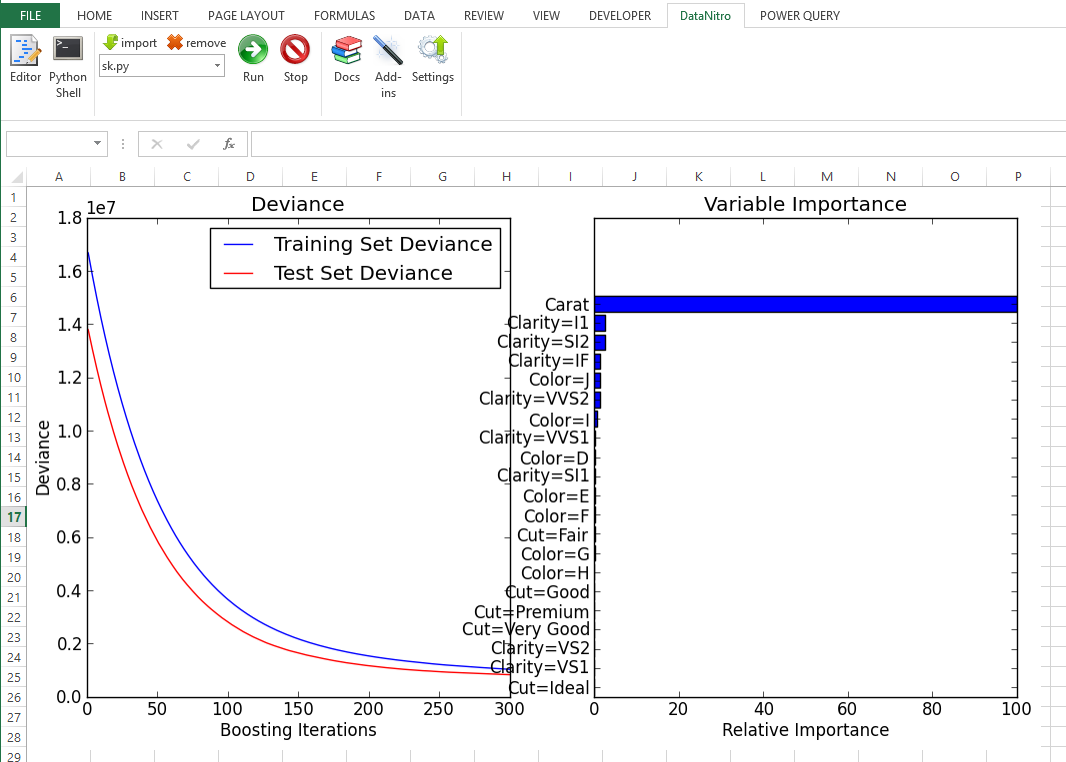
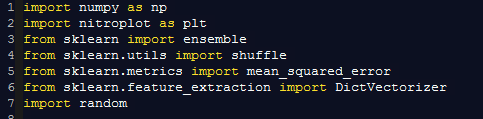
(This means the “cut” plot is five-dimensional, and the others are four-dimensional.)

This conveys more data than a regular scatter plot. For example, the rightmost plot shows that the most valuable diamonds have the worst clarity. This would be confusing on a normal plot; here, we can tell that the underlying reason is the size of the stones: bigger diamonds are more valuable, pretty much without regard for clarity. (*This is the main point of this section; if the audience seems confused, try to explain it very clearly and with references to the chart.)*

1. Interactive Predictive Model  
     
   **Goal:** Build an interactive dashboard.  
     
   **Instructions:** Select “predict.py” in the the datanitro toolbar’s dropdown, and press “Run”.  
     
   Type in a few values under “carat” as input: 0.25, 0.99, 1.0, 2.5, 5.0. Pause as you enter each number so the audience can see the script working. Ask the audience to select a few values. (There’s only basic error-checking, so don’t input non-numbers.) **Talking Points:**  
   DataNitro doesn’t halt Excel as it runs. This lets you turn Excel into a dashboard, with a Python script as the backend. This example is using an extremely basic algorithm to predict diamond price based on carat weight.

**Instructions (2):**Press “Editor” to bring up the predict.py script, and point out lines 37 to 43.  
  
  
**Talking Points (2):**  
You can write an interactive script just by using a while loop – you don’t need anything more complicated. The script will run in the background as users work with the spreadsheet.

(Close the predict.py shell window)

1. Scikit Learn  
   **Goal:** Demonstrate Machine Learning in Excel  
     
   **Instructions:**  
   Select “sk.py” in the datanitro toolbar’s dropdown, and press “Run”.  
     
   The following chart should appear:  
     
     
     
   **Talking Points:**If we want to build a more sophisticated model, we can use SciKit Learn – Python’s Machine Learning library. Here’s an example of applying a Gradient Boosting regression model to the dataset.  
     
   From the graph on the right, we can see that ignoring everything but the carat weight isn’t a bad approximation – it’s basically all that matters for determining price.  
     
   If we wanted to, we could set up another interactive dashboard for price prediction using an ML model.  
     
   **Instructions (2):** Press “Editor” to bring up the sk.py script  
     
   Point out lines 1 through 7 in the script:  
     
     
   **Talking Points (2):**(Point out the imports in the code)  
     
   DataNitro works with regular CPython, so you can use any Python library directly in Excel. This demo is using the Anaconda Python distribution, which includes Numpy, Scipy, Pandas, SciKit Learn (used here), and a host of other scientific packages.   
     
   In addition to numerical and statistical packages, Python also includes a number of libraries for connecting to other resources, such as pyodbc for easy database integration and requests for working with web resources.  
     
   (Close the editor.)
2. Wrap-up  
     
   People can download this code at [www.datanitro.com/diamonds.zip](http://www.datanitro.com/diamonds.zip) and see more examples at [www.datanitro.com/blog/](http://www.datanitro.com/blog/). The software is free for students, university researchers, and personal projects.  
     
   If someone wants to try it in a commercial setting, I can give them an extended trial – tell them to contact me directly at [ben@datanitro.com](mailto:ben@datanitro.com).