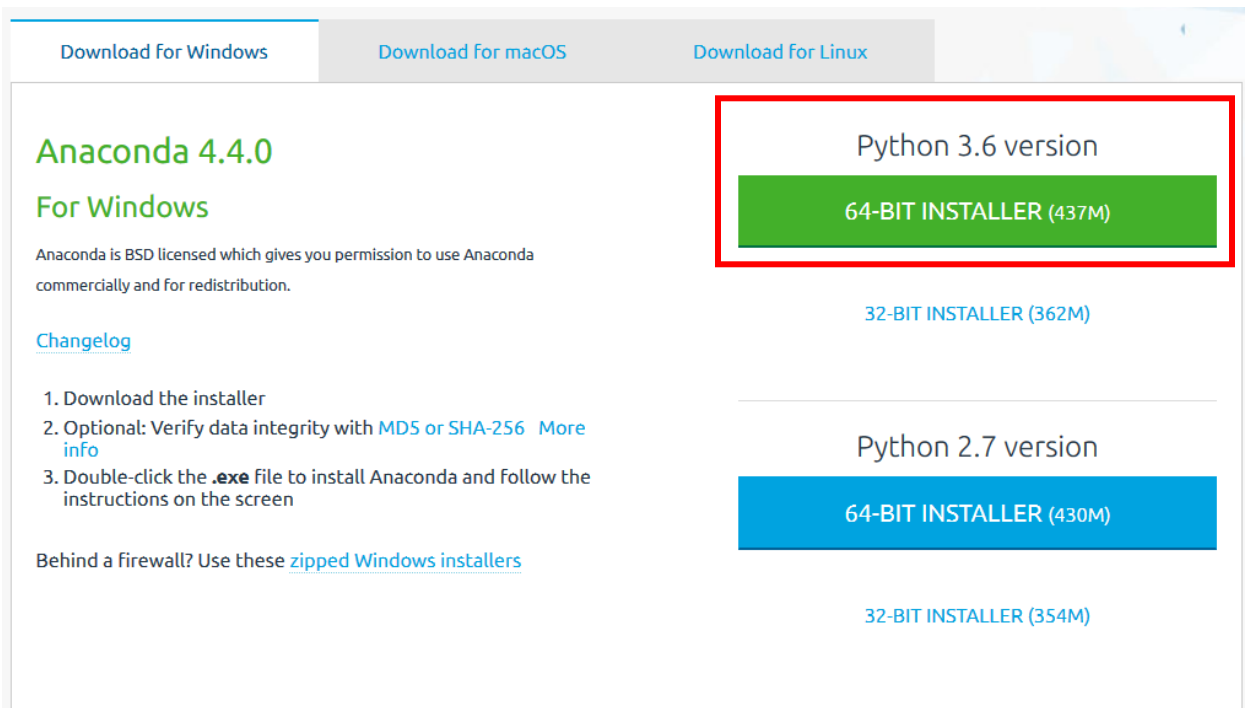


Getting Started with Python and Jupyter Notebook

Downloading

The simplest way to download everything you need for basic to intermediate Python applications is to download Anaconda. Anaconda automatically installs Python and its most common packages, as well as the Jupyter Notebook and Spyder interfaces to run Python programs.

To download Anaconda, visit: <https://www.continuum.io/downloads>



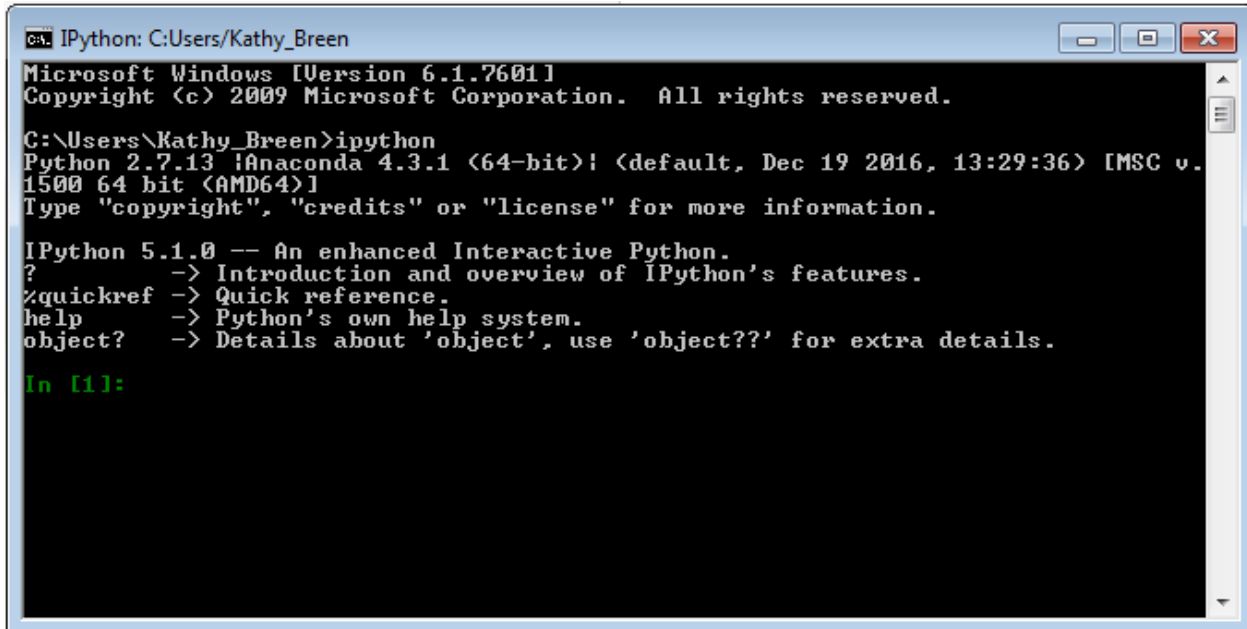
The screenshot shows the Anaconda download page for Windows. At the top, there are three tabs: "Download for Windows", "Download for macOS", and "Download for Linux". The "Download for Windows" tab is selected. Below the tabs, the text "Anaconda 4.4.0 For Windows" is displayed. To the right, there are two main sections for Python versions. The first section is for "Python 3.6 version" and contains a green button labeled "64-BIT INSTALLER (437M)". Below this button is a link for "32-BIT INSTALLER (362M)". The second section is for "Python 2.7 version" and contains a blue button labeled "64-BIT INSTALLER (430M)". Below this button is a link for "32-BIT INSTALLER (354M)". On the left side, under the "Anaconda 4.4.0 For Windows" heading, there is a paragraph stating "Anaconda is BSD licensed which gives you permission to use Anaconda commercially and for redistribution." followed by a link to the "Changelog". Below the changelog, there is a list of three steps: 1. Download the installer, 2. Optional: Verify data integrity with MD5 or SHA-256 (with a link to "More info"), and 3. Double-click the .exe file to install Anaconda and follow the instructions on the screen. At the bottom left, there is a link for "Behind a firewall? Use these zipped Windows installers".

Python 2.x is the legacy version – it's been around for a while and is tried and true. Python 3.x versions are stable development versions which, while they may have the occasional bug, are the basis for future developments in the language. Most deep learning frameworks require 3.x.

To view a list of **150 packages** automatically installed by Anaconda or view documentation, visit: <https://docs.continuum.io/anaconda/>

Running Python

Once you've installed Anaconda, you're ready to write and run some code. You have several options to run Python code, but the most common are iPython, Spyder, and Jupyter Notebooks. If you prefer command-line processing, by all means use iPython. Just open up a command window, and type `ipython`. If you're running *.py files you've previously saved, you'll want to navigate to that directory first. Once you initiate iPython, your command window should look like this:

A screenshot of a Windows command prompt window titled "C:\Users\Kathy_Breen". The window shows the output of running the "ipython" command. The text is color-coded: "Microsoft Windows [Version 6.1.7601] Copyright (c) 2009 Microsoft Corporation. All rights reserved." is in white; "C:\Users\Kathy_Breen>ipython" is in white; "Python 2.7.13 |Anaconda 4.3.1 (64-bit)| (default, Dec 19 2016, 13:29:36) [MSC v.1500 64 bit (AMD64)]" is in green; "Type 'copyright', 'credits' or 'license' for more information." is in white; "IPython 5.1.0 -- An enhanced Interactive Python." is in white; the help text ("? -> 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.") is in white; and "In [1]:" is in green. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

```
C:\Users\Kathy_Breen
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Kathy_Breen>ipython
Python 2.7.13 |Anaconda 4.3.1 (64-bit)| (default, Dec 19 2016, 13:29:36) [MSC v.
1500 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 5.1.0 -- 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]:
```

You'll notice that now when you enter text, it will appear color-coded. Colors differentiate between types of inputs and outputs, which helps you keep track of what you're typing. iPython is especially useful as a quick reference and help tool, and it's worth familiarizing yourself with basic usage; however, many people find it more advantageous to work with Spyder or Jupyter Notebook.

If you're familiar with workspace IDEs for Matlab or R (Rstudio), you may prefer to work in Spyder (Scientific **P**ython **D**evelopment **E**nvi**R**onment). To start Spyder, open Anaconda Navigator from the Start menu. Once the navigator window opens, launch Spyder. The Spyder GUI has the advantage of displaying workspace variables in the "variable explorer" window, which means that you can view the name, type, size, and content of all assigned data.

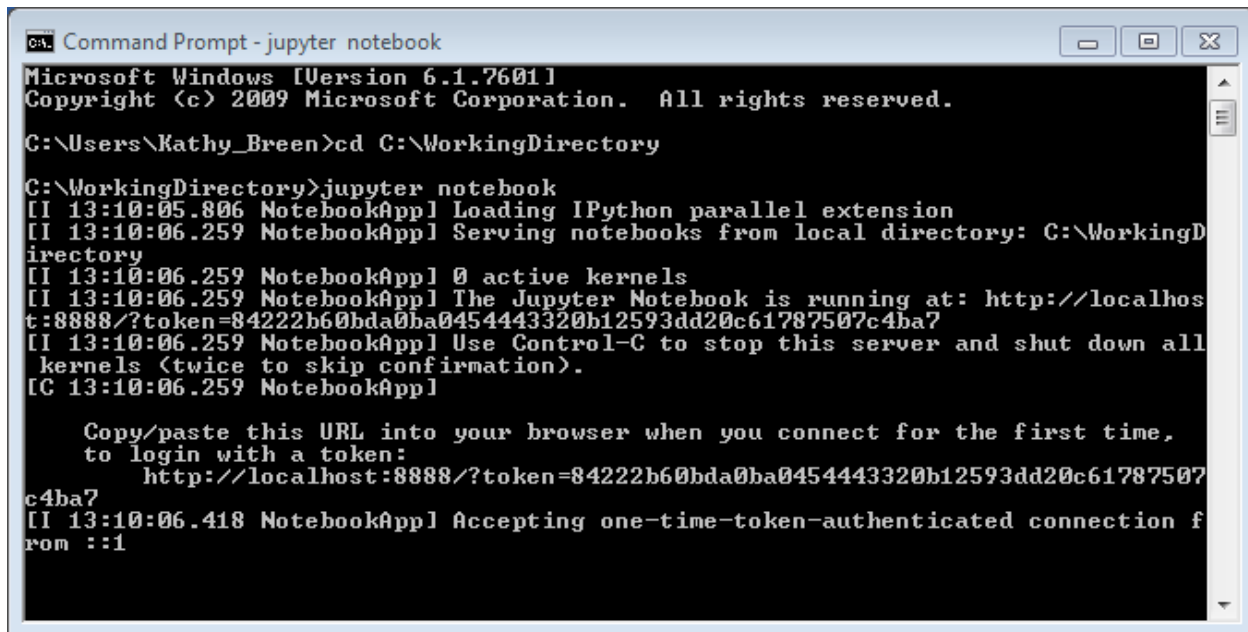
Jupyter Notebook (**J**ulia **P**ython and **R**) is a frontend terminal that uses iPython kernels to process code. A full explanation can be found here:

http://jupyter.readthedocs.io/en/latest/architecture/how_jupyter_ipython_work.html

Think of a terminal as an interface that displays inputs and outputs. It's where you type and edit your code. A kernel is like an engine to run your code. When you run your code for the first time, you initialize a kernel process. This means that any variables you define will remain in memory if you re-run your code. Once you close Jupyter Notebook, your kernel process is terminated.

To open Jupyter Notebook, first open a command window and navigate to the folder where you want your working directory to be, then type `jupyter notebook`.

A working directory is like home base. Some people put all their Python scripts in one folder, other people save all files related to a particular project in a folder. Your choice.



```
Command Prompt - jupyter notebook
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Kathy_Breen>cd C:\WorkingDirectory

C:\WorkingDirectory>jupyter notebook
[I 13:10:05.806 NotebookApp] Loading IPython parallel extension
[I 13:10:06.259 NotebookApp] Serving notebooks from local directory: C:\WorkingD
irectory
[I 13:10:06.259 NotebookApp] 0 active kernels
[I 13:10:06.259 NotebookApp] The Jupyter Notebook is running at: http://localhos
t:8888/?token=84222b60bda0ba0454443320b12593dd20c61787507c4ba7
[I 13:10:06.259 NotebookApp] Use Control-C to stop this server and shut down all
kernels (twice to skip confirmation).
[CG 13:10:06.259 NotebookApp]

Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
http://localhost:8888/?token=84222b60bda0ba0454443320b12593dd20c61787507
c4ba7
[I 13:10:06.418 NotebookApp] Accepting one-time-token-authenticated connection f
rom ::1
```

After a minute or two, a “Home” Notebook tab will open in your default internet browser via a local host. A “local host” means that the code is running directly from your computer, not using an internet connection, and uses the browser for display. On the Home tab, you’ll see a list of all the files in your working directory, including your existing notebooks. To open an existing notebook, just click on the name in your list. To open a new notebook, click “New” > “Python x”, and a new notebook will open in a separate browser tab. **Do not close the command window until you are finished with Jupyter – if you do, you will shut down the kernel supporting your local host.** You can also launch Jupyter Notebooks from Anaconda Navigator, but doing so will not give you access to files stored in a directory higher than your Documents folder or any external hardware. The preferred method is to launch Jupyter Notebooks via a command interface.

Import Packages

Now that you’re up and running, you need to import packages. Some of the most common packages you will need are **numpy**, **pandas**, **matplotlib**, and **os** (<https://docs.continuum.io/anaconda/packages/pkg-docs>).

Notebooks operate in a cell structure, where only the code in the active cell is run. A cell is active if it is outlined in either blue or green. A blue outline means the cell is selected, a green outline means the cell is selected and in edit mode. It’s a good idea to import all of your packages in the first cell, run it, and write the rest of your code in the following cells. There are several ways to import a package, like this:

```
# import a package
import packagename

# import a package using a short name, so that you don't have to type
the long name every time you want to use it (EX: import numpy as np)
import packagename as shortname

# import a single or select modules from a package. Some packages are
BIG, so if you're confident you only need a few modules, you can
import those only
import module from packagename
```

A python **package** is a collection of **modules** which perform a particular class of functions. Modules are simply *.py files. The base package (aka standard library) for Python includes functions to perform common tasks like the calculation of summary statistics, min/max, etc. If you want to use Python capabilities not included in the base package, you will need to import the necessary package or individual modules. Think of packages like toolboxes and modules as tools. Some packages have all the tools you need for document tree navigation, others for regular expressions, advanced data processing, etc. If all you need is a single or handful of modules, just import it from its respective package without having to load unnecessary modules. If you don't know what you need, take it all.

How to decide which packages to use? Google. Documentation. User Forums.

If you're not sure how to code something, here's a simple Google search format to get you started:

"program [version] problem in as few words as possible"

Examples:

"python 2.7 plot data"

"python 2.7 concatenate strings"

"jupyter notebook display data frame"

The last example will show search results for the Pandas package, including user forums and documentation (<https://pandas.pydata.org/pandas-docs/stable/options.html>). **Documentation** explains the usage, options, and inputs for package modules. **User forums**, such as StackExchange or GoogleGroups, are online spaces for community-supplied answers and fixes to common issues. More often than not, a problem you're having has been solved before with a variety of fixes, so always make sure to search the forums before asking a new question. If you've never used forums before, get a quick introduction here: <https://stackoverflow.com/tour>.

Resources

If you're new to Python (or any programming language), it's a good idea to get a grasp of the syntax and basic operations before trying to write your own code. Luckily, there are numerous free resources you can turn to.

- **Python.org:** Documentation, tutorials, code examples, news, etc...
- **Codecademy.com:** Free starter courses for multiple open source languages like Python, JavaScript, SQL, etc. This is a great way to get started or to refresh your memory if it's been a while.
- **Stackoverflow.com:** This is the most common user forum that will come up in your search results. This forum is part of a wider group under the StackExchange umbrella, which houses forums a variety of topics. Stackoverflow is the programming forum, and you can get just about any programming curiosity answered there. If you plan on asking questions, do your best to research the issue first and read through answers to similar issues.
- **Python Cookbook:** Beazley, D., & Jones, B.K. (2013). *Python Cookbook, Third Edition*. O'Reilly Media.