iPython Supplement for USGS

John Strickler

Version 1.0, May 2023

Table of Contents

Chapter 1: IPython and Jupyter

Objectives

- Learn the basics of IPython
- Apply magics
- List and replay commands
- Run external commands
- Create profiles
- Use Jupyter notebooks

About IPython

- Enhanced python interpreter
- Great for "playing around" with Python
- Saves running entire script
- Not intended for application development
- Embedded in Jupyter notebooks

IPython is an enhanced interpreter for Python. It provides a large number of "creature comforts" for the user, such as name completion and improved help features.

It is very handy for quickly trying out Python features or for casual data analysis.

Command line interface

When started from a command line, starts a read-execute-print loop (REPL), also known as an interactive interpreter.

Ipython uses different colors for variables, functions, strings, comments, and so forth.

Jupyter notebook

The most flexible and powerful way to run IPython is embedded in a Jupyter notebook. This mode starts a dedicated web server and begins a session using your default web browser. From the home page of Jupyter, you can create a Jupyter notebook containing Python code.

Starting IPython

- Type `ipython` at the command line
- Huge number of options

To get started with IPython

- Type `ipython` at the command line

OR

- Double-click the IPython icon from Windows explorer.

IPython works like the normal interactive Python interpreter, but with many more features.

There is a huge number of options. To see them all, invoke IPython with the --help-all option:

```
ipython —help-all
```

TIP Use the `--colors=NoColor` option to turn off syntax highlighting and other colorized features.

Getting Help

- ? basic help
- %quickref quick reference
- help standard Python help
- thing? help on thing

IPython provides help in several ways.

Typing `?` at the prompt will display an introduction to IPython and a feature overview.

For a quick reference card, type `%quickref`.

To start Python’s normal help system, type `help`.

For help on any Python object, type `object?` or `?object`. This is similar to saying `help("object")` in the default interpreter, but is "smarter".

TIP For more help, add a second question mark. This does not work for all objects, however, and sometimes it displays the source code of the module containing the object definition.

IPython features

- Name completion (variables, modules, methods, folders, files, etc.)
- Enhanced help system
- Autoindent
- Syntax highlighting
- 'Magic' commands for controlling IPython itself
- Easy access to shell commands
- Dynamic introspection (dir() on steroids)
- Search namespaces with wildcards
- Commands are numbered (and persistent) for recall
- Aliasing system for interpreter commands
- Simplified (and lightweight) persistence
- Session logging (can be saved as scripts)
- Detailed tracebacks when errors occur
- Session restoring (playback log to specific state)
- Flexible configuration system
- Easy access to Python debugger
- Simple profiling
- Interactive parallel computing (if supported by hardware)
- Background execution in separate thread
- Auto-parentheses ('sin 3' becomes 'sin(3)'
- Auto-quoting (',foo a b' becomes 'foo("a","b")'

Tab Completion

- Press `Tab` to complete
  - keywords
  - variables
  - modules
  - methods and attributes
  - parameters to functions
  - file and directory names

Pressing `Tab` will invoke **tab completion**, AKA **autocomplete**. If there is only one possible completion, it will be expanded. If there is more than one completion that will match, IPython will display a list of possible completions.

Autocomplete works on keywords, functions, classes, methods, and object attributes, as well as paths from your file system.

© 2023 CJ Associates *(Licensed to Protech)*

Magic Commands

- Start with `%` (line magic) or `%%` (cell magic)
- Simplify common tasks
- Use `%lsmagic` to list all magic commands

One of the enhancements in IPython is the set of "magic" commands. These are meta-commands (macros) that help you manipulate the IPython environment.

Normal magics apply to a single line. Cell magics apply to a cell (a group of lines).

For instance, `%history` will list previous commands.

Type `lsmagic` for a list of all magics

TIP If the magic command is not the same as a name in your Python code, you can leave off the leading `%` or `%%`.

Loading and running Python scripts

- Run script in current session
- `%run` runs script
- `%load` loads script source code into IPython

IPython provides two magics to run scripts — one to run directly, and one to run indirectly. Both will run the script in the context of the current IPython session.

Running scripts directly

The `%run` magic just takes a script name, and runs it. This method does not allow IPython magics to be executed as part of a script.

```
In [1]: %run ../EXAMPLES/my_vars.py
```

```
In [2]:  user_name
Out[2]: 'Susan'
```

```
In [3]: snake
Out[3]: 'Eastern Racer'
```

© 2023 CJ Associates *(Licensed to Protech)*

Running scripts indirectly

The `%load` magic takes a script name, and loads the contents of the script so it can then be executed.

This method allows IPython magics to be executed as part of a script.

This also useful if you want to run a script, but edit the script before it is run.

```
In [4]: %load imports.py
```

```
In [5]: # %load imports.py
...: import numpy as np
...: import scipy as sp
...: import pandas as pd
...: import matplotlib.pyplot as plt
...: import matplotlib as mpl
...: %matplotlib inline
...: import seaborn as sns
...: sns.set()
...:
...:
```

External commands

- Precede command with `!`
- Can assign output to variable

Any OS command can be run by starting it with a `!`.

The resulting output is returned as a list of strings (stripping the trailing `\n` characters). The result can be assigned to a variable.

Windows

```
In [3]: !dir DATA\*.csv
 Volume in drive Z is Shared Folders
 Volume Serial Number is 0000-0064

 Directory of Z:\Desktop\py2forsci\DATA

02/20/2014  01:53 PM             5,511 airport_boardings.csv
02/20/2014  01:53 PM             2,182 energy_use_quad.csv
02/20/2014  01:53 PM             4,993 parasite_data.csv
               3 File(s)         12,686 bytes
               0 Dir(s)  352,625,324,032 bytes free

In [4]:
```

Non-Windows (Linux, OS X, etc)

```
In [2]: !ls -l DATA/*.csv
-rwxr-xr-x  1 jstrick  staff  5511 Jan 27 19:44 DATA/airport_boardings.csv
-rwxr-xr-x  1 jstrick  staff  2182 Jan 27 19:44 DATA/energy_use_quad.csv
-rwxr-xr-x  1 jstrick  staff  4642 Jan 27 19:44 DATA/parasite_data.csv

In [3]:
```

© 2023 CJ Associates *(Licensed to Protech)*

Using history

- use %history magic
- `history` *list commands*
- `history -n` *list commands with numbers*
- `hist` *shortcut for "history"*

The `%history` magic will list previous commands. Use `-n` to list commands with their numbers.

Selecting commands

You can select a single command or a range of commands separated by a dash.

```
history 5
history 6-10
```

Use `~N/`, where N is 1 or greater, to select commands from previous sessions.

```
history ~2/3  third command in second previous session
```

To select more than one range or individual command, separate them by spaces.

```
history 4-6 9 12-16
```

TIP The same syntax can be used with `%edit`, `%rerun`, `%recall`, `%macro`, `%save` and `%pastebin`.

Recalling commands

The `%recall` magic will recall a previous command by number. It will leave the cursor at the end of the command so you can edit it.

```
    recall 12
    recall 4-7
```

Rerunning commands

`%rerun` will re-run a previous command without waiting for you to press `Enter`.

© 2023 CJ Associates *(Licensed to Protech)*

Saving sessions

- Save commands to Python script
- Specify one or more commands
- Use `%save` magic

It is easy to save a command, a range of commands, or any combination of commands to a Python script using the `%save` magic.

The syntax is

```
%save filename selected commands
```

.py will be appended to the filename.

Using Pastebin

- Online "clipboard"
- Use `%pastebin` command

Pastebin is a free online service that accepts pasted text and provides a link to access the text. It can be used to share code snippets with other programmers.

The `%pastebin` magic will paste selected commands to **Pastebin** and return a link that can be used to retrieve them. The link provided will expire in 7 days.

Use `-d` to specify a title for the pasted code.

```
link = %pastebin -d "my code" 10-15  write commands 10 through 15 to Pastebin and get
link
```

TIP Add ".txt" to the link to retrieve the plain text that you pasted. This can be done with `requests`:

```python
import requests
link = %pastebin -d "my code" 10-15
pasted_text = requests.get(link + '.txt').text
```

© 2023 CJ Associates *(Licensed to Protech)*

Benchmarking

- Use %timeit

IPython has a handy magic for benchmarking.

```
In [1]: color_values = { 'red':66, 'green':85, 'blue':77 }

In [2]: %timeit red_value = color_values['red']
10000000 loops, best of 3: 54.5 ns per loop

In [3]: %timeit red_value = color_values.get('red')
10000000 loops, best of 3: 115 ns per loop
```

%timeit will benchmark whatever code comes after it on the same line. %%timeit will benchmark contents of a notebook cell

Profiles

- Stored in `.ipython` folder in home folder
- Contains profiles and other configuration
- Can have multiple profiles
- `ipython profile` subcommands
  - `list`
  - `create`
  - `locate`

IPython supports *profiles* for storing custom configurations and startup scripts. There is a default profile, and any number of custom profiles can be created.

Each profile is a separate subfolder under the `.ipython` folder in a users's home folder.

Creating profiles

Use `ipython profile create name` to create a new named profile. If `name` is omitted, this will create the default profile (if it does not already exist)

Listing profiles

Use `ipython profile list` to list all profiles

Finding profiles

`ipython profile locate name` will display the path to the specified profile. As with creating, omitting the name shows the path to the default profile.

© 2023 CJ Associates *(Licensed to Protech)*

```
.ipython
├── cython
│   └── Users
│       └── mikedev
├── extensions
├── nbextensions
├── profile_default
│   ├── db
│   ├── ipython_config.py
│   ├── ipython_kernel_config.py
│   ├── log
│   ├── pid
│   ├── security
│   ├── startup
│   │   ├── 00_imports.py
│   │   └── 10_macros.py
│   └── static
│       └── custom
└── profile_science
    ├── db
    ├── ipython_config.py
    ├── ipython_kernel_config.py
    ├── log
    ├── pid
    ├── security
    └── startup
        └── 00_imports.py
```

Configuration

IPython has many configuration settings. You can change these settings by creating or editing the script named `ipython_config.py` in a profile folder.

Within this script you can use the global config object, named `c`.

For instance, the line

```
c.InteractiveShellApp.pylab_import_all = False
```

Will change how the `%pylab` magic works. When true, it will populate the user namespace with the contents of `numpy` and `pylab` as though you had entered `from numpy import *` and `from pylab import *`

When false, it will just import `numpy` as `np` and `pylab` as `pylab`.

Link to all IPython settings:

https://ipython.org/ipython-doc/3/config/options/index.html

Note that there are four groups of settings.

TIP When you create a profile, this config script is created with some commented code to get you started.

© 2023 CJ Associates *(Licensed to Protech)*

Startup

Startup scripts allow you to execute frequently used code, especially imports, when starting IPython.

Startup scripts go in the `startup` folder of the profile folder. All Python scripts in this folder will be executed, in lexicographical (sorted) order.

The scripts will be executed in the context of the IPython session, so all imports, variables, functions, classes, and other definitions will be available in the session.

TIP It is convenient to prefix the startup scripts with "00", "10", "20", and so forth, to set the order of execution.

Jupyter notebooks

- Extension of IPython
- Puts the interpreter in a web browser
- Code is grouped into "cells"
- Cells can be edited, repeated, etc.

In 2015, the developers of IPython pulled the notebook feature out of IPython to make a separate product called Jupyter. It is still invoked via the `jupyter notebook` command, and now supports over 130 language kernels in addition to Python.

A Jupyter notebook is a journal-like python interpreter that lives in a browser window. Code is grouped into cells, which can contain multiple statements. Cells can be edited, repeated, rearranged, and otherwise manipulated.

A notebook (i.e, a set of cells, can be saved, and reopened). Notebooks can be shared among members of a team via the notebook server which is built into Jupyter.

Jupyter Notebook Demo

At this point please start the Jupyter notebook server and follow along with a demo of Jupyter notebooks as directed by the instructor

Open an Anaconda prompt and navigate to the top folder of the student files, then

```
cd NOTEBOOKS
jupyter notebook
```

© 2023 CJ Associates *(Licensed to Protech)*

For more information

- https://ipythonbook.com

Appendix A: Where do I go from here?

Resources for learning Python

These are from Jessica Garson, who, among other things, teaches Python classes at NYU. (Used with permission).

Run the script **where_do_i_go.py** to display a web page with live links.

Resources for Learning Python

Just getting started

Here are some resources that can help you get started learning how to code.

- Code Newbie Podcast
- Dive into Python3
- Learn Python the Hard Way
- Learn Python the Hard Way
- Automate the Boring Stuff with Python
- Automate the Boring Stuff with Python

So you want to be a data scientist?

- Data Wrangling with Python
- Data Analysis in Python
- Titanic: Machine Learning from Disaster
- Deep Learning with Python
- How to do X with Python
- Machine Learning: A Probabilistic Prospective

So you want to write code for the web?

- Learn flask, some great resources are listed here
- Django Polls Tutorial
- Hello Web App
- Hello Web App Intermediate
- Test-Driven-Development for Web Programming
- 2 Scoops of Django

- HTML and CSS: Design and Build Websites
- JavaScript and JQuery

Not sure yet, that's okay!

Here are some resources for self guided learning. I recommend trying to be very good at Python and the rest should figure itself out in time.

- Python 3 Crash Course
- Base CS Podcast
- Writing Idiomatic Python
- Fluent Python
- Pro Python
- Refactoring
- Clean Code
- Write music with Python, since that's my favorite way to learn a new language

Appendix B: Python Bibliography

Data Science

- ***Building machine learning systems with Python***. William Richert, Luis Pedro Coelho. Packt Publishing
- ***High Performance Python***. Mischa Gorlelick and Ian Ozsvald. O'Reilly Media
- ***Introduction to Machine Learning with Python***. Sarah Guido. O'Reilly & Assoc.
- ***iPython Interactive Computing and Visualization Cookbook***. Cyril Rossant. Packt Publishing
- ***Learning iPython for Interactive Computing and Visualization***. Cyril Rossant. Packt Publishing
- ***Learning Pandas***. Michael Heydt. Packt Publishing
- ***Learning scikit-learn: Machine Learning in Python***. Raúl Garreta, Guillermo Moncecchi. Packt Publishing
- ***Mastering Machine Learning with Scikit-learn***. Gavin Hackeling. Packt Publishing
- ***Matplotlib for Python Developers***.Sandro Tosi.Packt Publishing
- ***Numpy Beginner's Guide.Ivan Idris***.Packt Publishing
- ***Numpy Cookbook.Ivan Idris***.Packt Publishing
- ***Practical Data Science Cookbook.Tony Ojeda, Sean Patrick Murphy, Benjamin Bengfort, Abhijit Dasgupta***.Packt Publishing
- ***Python Text Processing with NLTK 2.0 Cookbook.Jacob Perkins***.Packt Publishing
- ***Scikit-learn cookbook.Trent Hauck***.Packt Publishing
- ***Python Data Visualization Cookbook.Igor Milovanovic***.Packt Publishing
- ***Python for Data Analysis.Wes McKinney.***. O'Reilly & Assoc

Design Patterns

- ***Design Patterns: Elements of Reusable Object-Oriented Software.Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides***.Addison-Wesley Professional
- ***Head First Design Patterns.Eric Freeman, Elisabeth Robson, Bert Bates, Kathy Sierra***.O'Reilly Media
- ***Learning Python Design Patterns.Gennadiy Zlobin***.Packt Publishing
- ***Mastering Python Design Patterns.Sakis Kasampalis***.Packt Publishing

General Python development

- ***Expert Python Programming.Tarek Ziadé***.Packt Publishing
- ***Fluent Python.Luciano Ramalho***. O'Reilly & Assoc.

- ***Learning Python, 2nd Ed..Mark Lutz, David Asher***. O'Reilly & Assoc.
- ***Mastering Object-oriented Python.Stephen F. Lott***.Packt Publishing
- ***Programming Python, 2nd Ed. .Mark Lutz***. O'Reilly & Assoc.
- ***Python 3 Object Oriented Programming.Dusty Phillips***.Packt Publishing
- ***Python Cookbook, 3rd. Ed.. David Beazley, Brian K. Jones***. O'Reilly & Assoc.
- ***Python Essential Reference, 4th. Ed..David M. Beazley***.Addison-Wesley Professional
- ***Python in a Nutshell.Alex Martelli***. O'Reilly & Assoc.
- ***Python Programming on Win32.Mark Hammond, Andy Robinson***. O'Reilly & Assoc.
- ***The Python Standard Library By Example.Doug Hellmann***.Addison-Wesley Professional

Misc

- ***Python Geospatial Development.Erik Westra***.Packt Publishing
- ***Python High Performance Programming.Gabriele Lanaro***.Packt Publishing

Networking

- ***Python Network Programming Cookbook.Dr. M. O. Faruque Sarker***.Packt Publishing
- ***Violent Python: A Cookbook for Hackers, Forensic Analysts, Penetration Testers and Security Engineers.T J O'Connor***.Syngress
- ***Web Scraping with Python.Ryan Mitchell***.O'Reilly & Assoc.

Testing

- ***Python Testing Cookbook.Greg L. Turnquist***.Packt Publishing
- ***Learning Python Testing.Daniel Arbuckle***.Packt Publishing
- ***Learning Selenium Testing Tools, 3rd Ed. .Raghavendra Prasad MG***.Packt Publishing

Web Development

- ***Building Web Applications with Flask.Italo Maia***.Packt Publishing
- ***Django 1.0 Website Development.Ayman Hourieh***.Packt Publishing
- ***Django 1.1 Testing and Development.Karen M. Tracey***.Packt Publishing
- ***Django By Example.Antonio Melé***.Packt Publishing
- ***Django Design Patterns and Best Practices.Arun Ravindran***.Packt Publishing
- ***Django Essentials.Samuel Dauzon***.Packt Publishing

- ***Django Project Blueprints.Asad Jibran Ahmed***.Packt Publishing
- ***Flask Blueprints.Joel Perras***.Packt Publishing
- ***Flask by Example.Gareth Dwyer***.Packt Publishing
- ***Flask Framework Cookbook.Shalabh Aggarwal***.Packt Publishing
- ***Flask Web Development.Miguel Grinberg***. O'Reilly & Assoc.
- ***Full Stack Python (e-book only).Matt Makai***.Gumroad (or free download)
- ***Full Stack Python Guide to Deployments (e-book only).Matt Makai***.Gumroad (or free download)
- ***High Performance Django.Peter Baumgartner, Yann Malet***.Lincoln Loop
- ***Instant Flask Web Development.Ron DuPlain***.Packt Publishing
- ***Learning Flask Framework.Matt Copperwaite, Charles O Leifer***.Packt Publishing
- ***Mastering Flask.Jack Stouffer***.Packt Publishing
- ***Two Scoops of Django: Best Practices for Django 1.11. Daniel Roy Greenfeld, Audrey Roy Greenfeld***.Two Scoops Press
- ***Web Development with Django Cookbook.Aidas Bendoraitis***.Packt Publishing

© 2023 CJ Associates *(Licensed to Protech)*

Index