[1]Contentscontents

Contents

1 Command Line Documentation

This is an example of generating .rst docs which shows the output from running ipython command line help.

2 Commentary

The basic help is way too long, the first help should just point people to the basic command and where they can get more detailed help.

3 IPython Help

Here is the basic ipython help available:

```
genericprompt$ ipython --help
genericoutput=========
 IPython
=========

Tools for Interactive Computing in Python
=========================================

    A Python shell with automatic history (input and output), dynamic object
    introspection, easier configuration, command completion, access to the
    system shell and more.  IPython can also be embedded in running programs.

Usage

    ipython [subcommand] [options] [-c cmd | -m mod | file] [--] [arg] ...

    If invoked with no options, it executes the file and exits, passing the
```

```
    remaining arguments to the script, just as if you had specified the same
    command with python. You may need to specify `--` before args to be passed
    to the script, to prevent IPython from attempting to parse them. If you
    specify the option `-i` before the filename, it will enter an interactive
    IPython session after running the script, rather than exiting. Files ending
    in .py will be treated as normal Python, but files ending in .ipy can
    contain special IPython syntax (magic commands, shell expansions, etc.).

    Almost all configuration in IPython is available via the command-line. Do
    `ipython --help-all` to see all available options.  For persistent
    configuration, look into your `ipython_config.py` configuration file for
    details.

    This file is typically installed in the `IPYTHONDIR` directory, and there
    is a separate configuration directory for each profile. The default profile
    directory will be located in $IPYTHONDIR/profile_default. For Linux users,
    IPYTHONDIR defaults to `$HOME/.config/ipython`, and for other Unix systems
    to `$HOME/.ipython`.  For Windows users, $HOME resolves to C:\Documents
    and Settings\YourUserName in most instances.

    To initialize a profile with the default configuration file, do::

      $> ipython profile create

    and start editing `IPYTHONDIR/profile_default/ipython_config.py`

    In IPython's documentation, we will refer to this directory as
    `IPYTHONDIR`, you can change its default location by creating an
    environment variable with this name and setting it to the desired path.

    For more information, see the manual available in HTML and PDF in your
    installation, or online at http://ipython.org/documentation.html.

Subcommands
-----------

Subcommands are launched as `ipython cmd [args]`. For information on using
subcommand 'cmd', do: `ipython cmd -h`.

locate
    print the path to the IPython dir
profile
    Create and manage IPython profiles.
console
```

```
    Launch the IPython terminal-based Console.
kernel
    Start a kernel without an attached frontend.
notebook
    Launch the IPython HTML Notebook Server.
nbconvert
    Convert notebooks to/from other formats.
qtconsole
    Launch the IPython Qt Console.
history
    Manage the IPython history database.

Options
-------

Arguments that take values are actually convenience aliases to full
Configurables, whose aliases are listed on the help line. For more information
on full configurables, see '--help-all'.

--no-autoindent
    Turn off autoindenting.
--autoedit-syntax
    Turn on auto editing of files with syntax errors.
--deep-reload
    Enable deep (recursive) reloading by default. IPython can use the
    deep_reload module which reloads changes in modules recursively (it
    replaces the reload() function, so you don't need to change anything to
    use it). deep_reload() forces a full reload of modules whose code may
    have changed, which the default reload() function does not.  When
    deep_reload is off, IPython will use the normal reload(), but
    deep_reload will still be available as dreload(). This feature is off
    by default [which means that you have both normal reload() and
    dreload()].
--confirm-exit
    Set to confirm when you try to exit IPython with an EOF (Control-D
    in Unix, Control-Z/Enter in Windows). By typing 'exit' or 'quit',
    you can force a direct exit without any confirmation.
--pylab
    Pre-load matplotlib and numpy for interactive use with
    the default matplotlib backend.
--matplotlib
    Configure matplotlib for interactive use with
    the default matplotlib backend.
--term-title
```

```
    Enable auto setting the terminal title.
--classic
    Gives IPython a similar feel to the classic Python prompt.
--autoindent
    Turn on autoindenting.
--no-automagic
    Turn off the auto calling of magic commands.
--banner
    Display a banner upon starting IPython.
--automagic
    Turn on the auto calling of magic commands. Type %%magic at the
    IPython  prompt  for  more information.
--no-deep-reload
    Disable deep (recursive) reloading by default.
--no-term-title
    Disable auto setting the terminal title.
--nosep
    Eliminate all spacing between prompts.
-i
    If running code from the command line, become interactive afterwards.
    Note: can also be given simply as '-i.'
--debug
    set log level to logging.DEBUG (maximize logging output)
--pprint
    Enable auto pretty printing of results.
--no-autoedit-syntax
    Turn off auto editing of files with syntax errors.
--quiet
    set log level to logging.CRITICAL (minimize logging output)
--no-color-info
    Disable using colors for info related things.
--color-info
    IPython can display information about objects via a set of func-
    tions, and optionally can use colors for this, syntax highlighting
    source code and various other elements.  However, because this
    information is passed through a pager (like 'less') and many pagers get
    confused with color codes, this option is off by default.  You can test
    it and turn it on permanently in your ipython_config.py file if it
    works for you.  Test it and turn it on permanently if it works with
    your system.  The magic function %%color_info allows you to toggle this
    interactively for testing.
--init
    Initialize profile with default config files.  This is equivalent
    to running `ipython profile create <profile>` prior to startup.
```

```
--no-pdb
    Disable auto calling the pdb debugger after every exception.
--quick
    Enable quick startup with no config files.
--no-confirm-exit
    Don't prompt the user when exiting.
--pydb
    Use the third party 'pydb' package as debugger, instead of pdb.
    Requires that pydb is installed.
--pdb
    Enable auto calling the pdb debugger after every exception.
--no-pprint
    Disable auto auto pretty printing of results.
--no-banner
    Don't display a banner upon starting IPython.
--profile=<Unicode> (BaseIPythonApplication.profile)
    Default: u'default'
    The IPython profile to use.
-c <Unicode> (InteractiveShellApp.code_to_run)
    Default: ''
    Execute the given command string.
--pylab=<CaselessStrEnum> (InteractiveShellApp.pylab)
    Default: None
    Choices: ['auto', 'gtk', 'inline', 'osx', 'qt', 'qt4', 'tk', 'wx']
    Pre-load matplotlib and numpy for interactive use, selecting a particular
    matplotlib backend and loop integration.
--autocall=<Enum> (InteractiveShell.autocall)
    Default: 0
    Choices: (0, 1, 2)
    Make IPython automatically call any callable object even if you didn't type
    explicit parentheses. For example, 'str 43' becomes 'str(43)' automatically
    The value can be '0' to disable the feature, '1' for 'smart' autocall, wher
    it is not applied if there are no more arguments on the line, and '2' for
    'full' autocall, where all callable objects are automatically called (even
    if no arguments are present).
--ipython-dir=<Unicode> (BaseIPythonApplication.ipython_dir)
    Default: u'/home/ana/.config/ipython'
    The name of the IPython directory. This directory is used for logging
    configuration (through profiles), history storage, etc. The default is
    usually $HOME/.ipython. This options can also be specified through the
    environment variable IPYTHONDIR.
--gui=<CaselessStrEnum> (InteractiveShellApp.gui)
    Default: None
    Choices: ('qt', 'wx', 'gtk', 'glut', 'pyglet', 'osx')
```

```
    Enable GUI event loop integration ('qt', 'wx', 'gtk', 'glut', 'pyglet',
    'osx').
--logappend=<Unicode> (InteractiveShell.logappend)
    Default: ''
    Start logging to the given file in append mode.
-m <Unicode> (InteractiveShellApp.module_to_run)
    Default: ''
    Run the module as a script.
--ext=<Unicode> (InteractiveShellApp.extra_extension)
    Default: ''
    dotted module name of an IPython extension to load.
--log-level=<Enum> (Application.log_level)
    Default: 30
    Choices: (0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN', 'ERROR', 'CRITICA
    Set the log level by value or name.
--colors=<CaselessStrEnum> (InteractiveShell.colors)
    Default: 'Linux'
    Choices: ('NoColor', 'LightBG', 'Linux')
    Set the color scheme (NoColor, Linux, or LightBG).
--matplotlib=<CaselessStrEnum> (InteractiveShellApp.matplotlib)
    Default: None
    Choices: ['auto', 'gtk', 'inline', 'osx', 'qt', 'qt4', 'tk', 'wx']
    Configure matplotlib for interactive use with the default matplotlib
    backend.
--cache-size=<Integer> (InteractiveShell.cache_size)
    Default: 1000
    Set the size of the output cache.  The default is 1000, you can change it
    permanently in your config file.  Setting it to 0 completely disables the
    caching system, and the minimum value accepted is 20 (if you provide a valu
    less than 20, it is reset to 0 and a warning is issued).  This limit is
    defined because otherwise you'll spend more time re-flushing a too small
    cache than working
--logfile=<Unicode> (InteractiveShell.logfile)
    Default: ''
    The name of the logfile to use.
--config=<Unicode> (BaseIPythonApplication.extra_config_file)
    Default: u''
    Path to an extra config file to load.
    If specified, load this config file in addition to any other IPython config

To see all available configurables, use `--help-all`

Examples
--------
```

```
ipython --pylab            # start in pylab mode
ipython --pylab=qt         # start in pylab mode with the qt4 backend
ipython --log-level=DEBUG  # set logging to DEBUG
ipython --profile=foo      # start with profile foo

ipython qtconsole          # start the qtconsole GUI application
ipython help qtconsole     # show the help for the qtconsole subcmd

ipython console            # start the terminal-based console application
ipython help console       # show the help for the console subcmd

ipython notebook           # start the IPython notebook
ipython help notebook      # show the help for the notebook subcmd

ipython profile create foo # create profile foo w/ default config files
ipython help profile       # show the help for the profile subcmd

ipython locate             # print the path to the IPython directory
ipython locate profile foo # print the path to the directory for profile ‘f

ipython nbconvert           # convert notebooks to/from other formats
```

4 nbconvert Help

Here is the help for the nbconvert tool:

```
genericprompt$ ipython nbconvert --help
genericoutputThis application is used to convert notebook files (*.ipynb) to va
formats.

Options
-------

Arguments that take values are actually convenience aliases to full
Configurables, whose aliases are listed on the help line. For more information
on full configurables, see '--help-all'.

--debug
    set log level to logging.DEBUG (maximize logging output)
--init
    Initialize profile with default config files.  This is equivalent
    to running ‘ipython profile create <profile>‘ prior to startup.
--quiet
```

```
    set log level to logging.CRITICAL (minimize logging output)
--stdout
    Write notebook output to stdout instead of files.
--profile=<Unicode> (BaseIPythonApplication.profile)
    Default: u'default'
    The IPython profile to use.
--notebooks=<List> (NbConvertApp.notebooks)
    Default: []
    List of notebooks to convert. Wildcards are supported. Filenames passed
    positionally will be added to the list.
--ipython-dir=<Unicode> (BaseIPythonApplication.ipython_dir)
    Default: u'/home/ana/.config/ipython'
    The name of the IPython directory. This directory is used for logging
    configuration (through profiles), history storage, etc. The default is
    usually $HOME/.ipython. This options can also be specified through the
    environment variable IPYTHONDIR.
--format=<CaselessStrEnum> (NbConvertApp.export_format)
    Default: 'full_html'
    Choices: ['basic_html', 'full_html', 'latex', 'markdown', 'python', 'reveal
    The export format to be used.
--writer=<DottedObjectName> (NbConvertApp.writer_class)
    Default: 'FilesWriter'
    Writer class used to write the  results of the conversion
--config=<Unicode> (BaseIPythonApplication.extra_config_file)
    Default: u''
    Path to an extra config file to load.
    If specified, load this config file in addition to any other IPython config
--log-level=<Enum> (Application.log_level)
    Default: 30
    Choices: (0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN', 'ERROR', 'CRITICA
    Set the log level by value or name.

To see all available configurables, use `--help-all`

Examples
--------

    The simplest way to use nbconvert is

    > ipython nbconvert mynotebook.ipynb

    which will convert mynotebook.ipynb to the default format (probably HTML).

    You can specify the export format with `--format`.
```

```
Options include ['basic_html', 'full_html', 'latex', 'markdown', 'python',

> ipython nbconvert --format latex mynotebook.ipnynb

You can also pipe the output to stdout, rather than a file

> ipython nbconvert mynotebook.ipynb --stdout

Multiple notebooks can be given at the command line in a couple of
different ways:

> ipython nbconvert notebook*.ipynb
> ipython nbconvert notebook1.ipynb notebook2.ipynb

or you can specify the notebooks list in a config file, containing::

    c.NbConvertApp.notebooks = ["my_notebook.ipynb"]

> ipython nbconvert --config mycfg.py
```