Chat Console Documentation

Release 0.1.*

Manfred Minimair

2018-07-27

Contents

Chat Console is a command console (shell) for Jupyter kernels, with enhancements for inline graphics, syntax highlighting, and much more. It facilitates collaborative work by supporting chat and displaying commands of multiple users asynchronously.

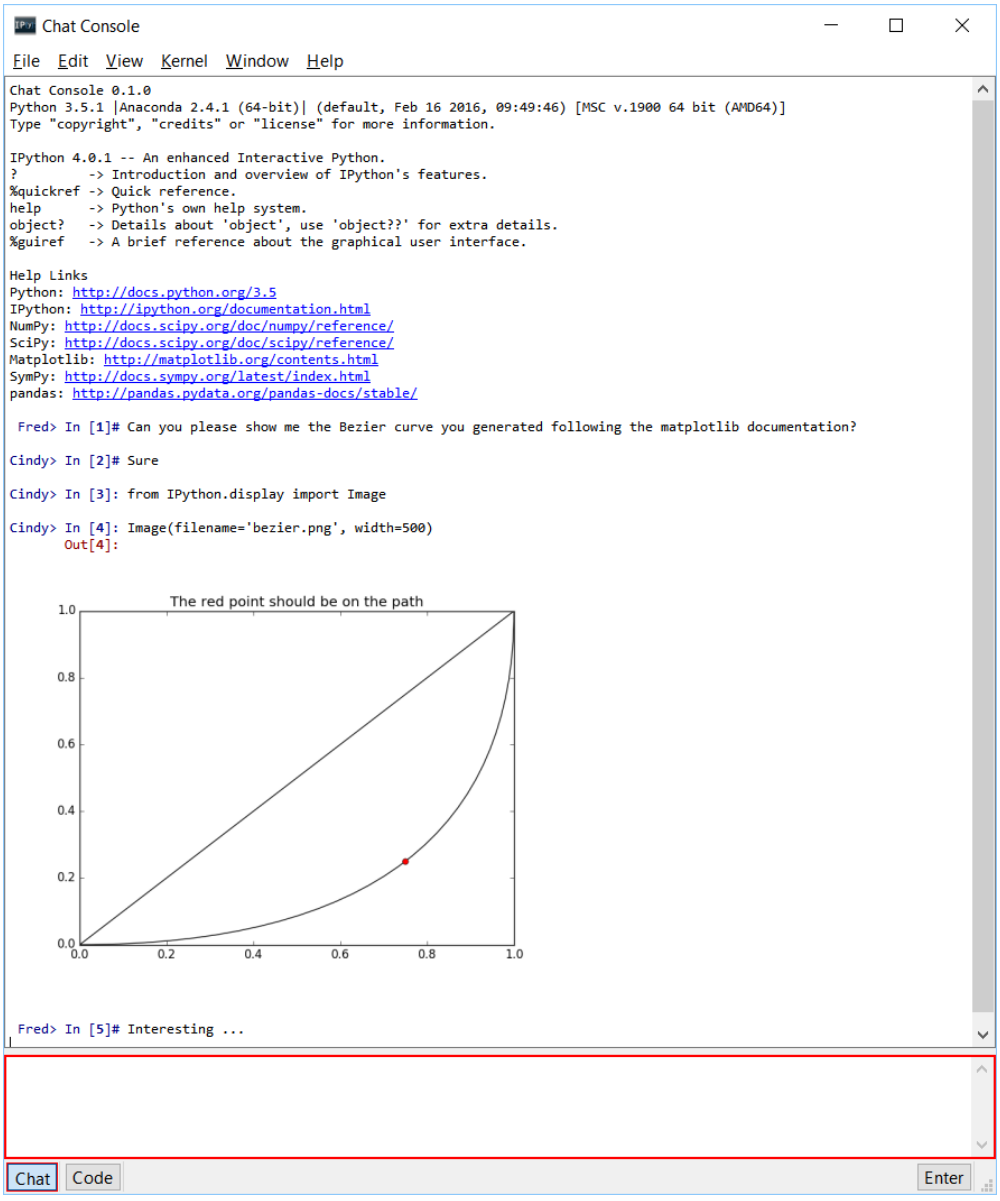
Contents

CHAPTER 1

Overview

Chat Console is a command console (shell) for Jupyter kernels, including IPython, with enhancements for inline graphics, syntax highlighting, and much more. It facilitates collaborative work by supporting chat and displaying commands of multiple users asynchronously.

The console can be used by a single user and offers special features when multiple users are connected to a kernel with multiple consoles. Each input is shown immediately in each users' console. Furthermore, the users can have shown the names of the users who provided the input. Additionally, the console can be switched between chat and command input mode. In chat input mode the user enters chat messages and in command input mode the user enters commands. Compare with the included image.


Chat Console
File Edit View Kernel Window Help
Chat Console 0.1.0
Python 3.5.1 |Anaconda 2.4.1 (64-bit)| (default, Feb 16 2016, 09:49:46) [MSC v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.
IPython 4.0.1 -- 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.
%guiref -> A brief reference about the graphical user interface.
Help Links
Python: http://docs.python.org/3.5
IPython: http://ipython.org/documentation.html
NumPy: http://docs.scipy.org/doc/numpy/reference/
SciPy: http://docs.scipy.org/doc/scipy/reference/
Matplotlib: http://matplotlib.org/contents.html
SymPy: http://docs.sympy.org/latest/index.html
pandas: http://pandas.pydata.org/pandas-docs/stable/
Fred> In [1]# Can you please show me the Bezier curve you generated following the matplotlib documentation?
Cindy> In [2]# Sure
Cindy> In [3]: from IPython.display import Image
Cindy> In [4]: Image(filename='bezier.png', width=500)
Out[4]:
The red point should be on the path
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Fred> In [5]# Interesting ...
Chat
Code
Enter

CHAPTER 2

Installation

2.1 Requirements

The Chat Console requires Qt, such as PyQt5, PyQt4, or PySide.

Note: Make sure that Qt is installed. Unfortunately, Qt cannot be installed using pip. The next section gives instructions on installing Qt.

2.2 Installing Qt (if needed)

We recommend installing PyQt with conda:

```
conda install pyqt
```

or with a system package manager. For Windows, PyQt binary packages may be used.

2.3 Install Chat Console using pip

To install:

```
pip install chconsole
```

CHAPTER 3

Configuration

The command line option –help-all shows all available configuration options available through the command line. Additionally the application can be configured through its configuration file. The configuration file can be generated by:

```
chconsole --generate-config
```

The configuration file is usually generated in the folder .jupyter located in the user's home directory. (For other possible locations of Jupyter configuration files see the corresponding Jupyter documentation)

CHAPTER 4

Usage

4.1 Launching the Chat Console and Connecting to Kernels

Start Chat Console with:

```
chconsole
```

or:

```
jupyter chconsole
```

After starting the console will ask the user to choose a connection file. The user may choose an existing file or cancel. If an existing file is chosen, the console will attach to a running Jupyter kernel specified by the file. Otherwise, the console will launch its own kernel.

The console tries to guess a suitable user name when starting, usually the user's login name. Having a unique user name is useful when several users collaborate while connected to one kernel. It is possible to specify the user name explicitly when launching the console with the user option:

```
chconsole --user <user name>
```

where <user name> stands for the user name to be used.

For convenience, the script:

```
chc-python
```

launches an IPython kernel to which consoles may connect. After starting the script, the user can choose an existing connection file or a new file name. If the connection file already exists then the script attempts to start a kernel with the specifications of the connection file. Otherwise, a new connection file will be created.

4.2 Sharing the Connection File

When multiple users connect to the same kernel, the consoles used for connecting need to have access to the connection file of the kernel. Therefore the connection file must be shared among the users. There are different ways of sharing. For example, the file can be distributed by e-mail. However, another convenient possibility is to create a shared network folder which is accessible to all users and where the kernel saves its connection file. The included script chc-python for starting the IPython kernel allows to browse for any accessible location to store the connection file. Alternatively, when the IPython kernel is launched on the command line with 'ipython kernel', the '-f' option allows to specify the connection file, including its desired path.

4.3 SSH Tunnels to Kernels

To establish connections between the kernel and the console, the kernel, that is, its ip address, must be accessible from the console. Sometimes, for example, if the kernel is behind a firewall, it may be necessary to create an ssh tunnel to the kernel. The Jupyter framework contains helpful options for creating such an ssh tunnel. See the documentation of Jupyter qtconsole for details on this process.

Connections through ssh can be configured in the configuration file as well as on the command line. In the configuration file, specify the required ssh key with JupyterConsoleApp.sshkey and the ip address of the ssh server with JupyterConsoleApp.sshserver. The ssh server can also be given on the command line with the option '–ssh'. If the user's login name on the client running the console differs from the login name on the ssh server, specify the server-side login name as part of the ip address: name @ ip address.

4.4 Connecting to a Sage Appliance

Sage is a mathematics software system that runs on an IPython kernel. It is possible to connect Chat Console, as any other Jupyter clients, to a running Sage notebook. Ensure that the Sage virtual machine accepts ssh login, as it is explained in the Sage documentation. As of Sage's version 7.0, after starting the notebook, the connection file of the notebook can be found in the folder /home/sage/.local/share/jupyter/runtime. Copy the connection file to the computer to run the console and establish a connection to the notebook through an ssh tunnel with login name 'sage'.

4.5 Chat and Command Entry

For command entry Chat Console provides all the features, users who know Jupyter qtconsole may expect. However, unlike qtconsole, all input is entered in a special input field at the bottom of the console's window. This feature allows showing inputs by other users immediately when they have been entered.

In addition, Chat Console supports chat communications among multiple users connected to the same kernel which qtconsole lacks. When the cursor is at the beginning of the entry field, hitting the tab key switches between chat and command entry modes. Chat messages can also be sent in command entry mode by starting them with a pound sign, such as:

```
# Hello, is there anybody?
```

Furthermore, the view menu provides an option which shows and hides the names of the users responsible for the corresponding chat and command inputs.

4.6 Special Key Strokes

When in the entry area, various key strokes have special effects.

Command entry

Enter Execute the text that has been entered

Shift-Enter Execute the entered text even if it may have syntax errors

Ctrl-Enter Enter a new line and do not execute

Alt-< Move the cursor to the start of the input field

Alt-> Move the cursor to the end of the input field

Alt-B Move the cursor to the start of the current word

Alt-F Move the cursor to the end of the current word

Alt-Backspace Delete from the cursor position to the beginning of the current word and save the deleted text in the kill ring

Alt-D, Alt-Delete or Ctrl-Delete Delete from the cursor position to the end of the current word and save the deleted text in the kill ring

Alt-Y Rotate the kill ring after retrieving the first entry from the kill ring with Ctrl-Y

Ctrl-I Interrupt the kernel

Ctrl-G Restart the kernel

Ctrl-O Release focus and move to the area where the commands and outputs are show

Ctrl-D or Delete Delete the current character at the cursor position

Ctrl-K Delete until the end of the line and store the deleted text in the kill ring

Ctrl-U Delete until the beginning of the line, clear any text selection and store the deleted text in the kill ring

Ctrl-Y Retrieve the last text saved in the kill ring

Ctrl-Backspace Delete until the beginning of the line and store the deleted text in the kill ring

Tab If the cursor is at the beginning of the entry field, switch to chat mode; otherwise try to complete the code contained in the entry field; otherwise insert spaces for indentation

Esc Clear the entry field

Chat entry

Most of the key strokes for navigation and editing from command entry are available. However, the kill ring is not used and the corresponding key strokes are disabled.