IPython and Vim

— Primer —

Jeonghyun Pyo

Korean Astronomy and Space Science Institute (KASI), KOREA

UST Astronomy Python Class
2012 March 20

1. **IPython**
   - Major Features (ver.0.12)
   - Launching IPython
   - Basic Usage
   - Qt Console
   - Web-based Notebook
   - Vim Integration

2. **Vim**
   - Python Interface
   - Basic Configuration for Python
   - Completion

IPython » Major Features (ver.0.12)

- Enhanced interactive Python shell
- Completion by typing `Tab` (keywords, methods, variables and files)
- Numbered input/output history (lists `In` and `Out`)
- “Magic” commands
- System shell access with `!`
- Automatic indentation
- Session logging & restoring
- Parallel computing
- Vim integration
- Debugger (pdb) access and profiler support
- Various frontends:
  Terminal console, Qt console, Browser-based notebook

IPython » Launching IPython

Terminal console

```
$ ipython [console]
```

Qt console

```
$ ipython qtconsole
```

Notebook

```
$ ipython notebook
```

IPython » Basic Usage

How to input codes

Terminal console

- Does not support multi-line input.
- Each line inputed (ended with `Enter`) are immediately executed.

Qt console

- Each line inputed (ended with `Enter`) are immediately executed.
- To input multi-line code,
  1. Hit `Ctrl` + `Enter` at the end of first line.
  2. Hit `Shift` + `Enter` at the end of last line.

IPython » Basic Usage

How to input codes

Notebook

- `Enter` does NOT execute codes, but continues input.
- `Shift` + `Enter`: Execute the current cell
- `Ctrl` + `Enter`: Execute the current cell in-place

IPython » Basic Usage

Tab completion

- Works for keywords, methods, variables and files

Automatic indentation

Examining objects

- `object_name?`
- Magic commands: `%pdoc`, `%pdef`, `%psource` and `%pfile`

Running and editing

- `%run`: Run Python script and load all of its data
  - `-t`: Timing the execution
  - `-d`: Run under the control of pdb debugger
  - `-p`: Run under the control of profiler
- `%edit`: Invoke external editor on the spot and execute the code

IPython » Basic Usage

History

- Navigate previous commands with ↑, ↓ and Ctrl+R
  - Or, in `vi` keymap, with `j` and `k`
- Input and output history are kept in `In` and `Out`.
- `%history`: Examine history
  - `-n`: Print line numbers
  - `-o`: Also print outputs
- Other magic commands using history
  `%edit`, `%rerun`, `%recall`, `%macro`, `%save` and `%pastebin`

IPython » Basic Usage

System shell commands

- Commands starting with `!` run at the system shell.
- Capture the output into a Python list: `files = !ls`
- Pass Python variables to system commands, e.g., `!grep -rF $pattern ipython/*`

Session logging and restoring

- Start `IPython` with `--logfile=foo.py`.
- Magic commands:
  - `%logstart [log_name [log_mode]]`: Activate logging
  - `%logoff` and `%logon`: Temporarily stop and resume loggin
  - `%logstop`: Deactivate logging

IPython » Basic Usage

Plotting with `matplotlib`

- Start `IPython` with
  - `--pylab`: Automatically detect a backend
  - `--pylab=backend`: Specify a backend (`tk`, `qt`, `wx`, `gtk`, `osx`)
- For inline plotting (Qt console or Notebook), start with `--pylab=inline`

IPython » Qt Console

Major features

- Inline figures

```
$ ipython qtconsole --pylab=inline
```

- Multi-line editing
  - To input multiple lines, use `Ctrl`+`Enter` and `Shift`+`Enter`.
- Syntax highlighting
- Graphical calltips
- `%loadpy`: Load a Python script

IPython » Web-based Notebook

Major features

- Inline figures

```
$ ipython notebook --pylab=inline
```

- Multi-line editing
  - To input multiple lines, just type in `Enter`.
  - To execute multiple lines, use `Ctrl`+`Enter` or `Shift`+`Enter`.
- Syntax highlighting
- Graphical calltips
- Compose text cells using HTML amd Markdown
- Import and export notebook documents (`.ipynb` and `.py`)

IPython » Vim Integration

How to start

1. Download `ipy.vim` from http://github.com/ivanov/vim-ipython.
2. Put `ipy.vim` into `~/.vim/ftplugin/python/`.
3. Fly a new Vim opening a Python code.
4. Fly an IPython kernel (or Qt console or notebook) and copy a line like

```
--existing kernel-27330.json
```

5. In Vim,

```
:IPythonClipboard
```

IPython » Vim Integration

How to use

- `Ctrl`+`S`
  Send a single line or a block.
- `F5`
  Run the file you are editing with IPython's `%run` magic.

Vim » Python Interface

- Refer to help:

```
:help python
```

- Execute Python statement(s):

```
:py[thon] {stmt}
:py[thon] << EOF
{script}
EOF
```

- Execute a Python script:

```
:pyf[ile] {file}
```

- Execute currently editing Python script:

```
:pyf[ile] %
```

Vim » Basic Configuration for Python

Edit `~/.vim/ftplugin/python.vim` and insert following lines

```
setlocal shiftwidth=4
setlocal tabstop=4
setlocal expandtab
setlocal smartindent
setlocal textwidth=79

" Settings for folding
setlocal foldmethod=indent
```

To use vi-like keymap in IPython, put following lines into `~/.inputrc`

```
set editing-mode vi
set keymap vi
```

Vim » Completion

In *raw* Vim

- Press `Ctrl`+`X` `Ctrl`+`O` in insert mode.
- Only one match → Automatically insert that item.
  Multiple matches → Navigate with `Ctrl`+`N` or `Ctrl`+`P` and press `Enter` to select

Vim » Completion

With `AutoComplPop`

How to install

1. Download `vim-autocomplpop.zip` from `http://www.vim.org/scripts/script.php?script_id=1879`
2. Unzip `vim-autocomplpop.zip` under `~/.vim/`
3. Fly a Vim and execute

```
:helptags ~/.vim/doc
:help acp
```

Vim » Completion

With `AutoComplPop`

How to use

- It automatically pop up a list of matched keywords.
- Add the following lines into `~/.vim/ftplugin/python.vim`

```
" To see preview window
let g:acp_completeoptPreview = 1
" Pop-up only more than 2 characters inputed
let g:acp_behaviorPythonOmniLength = 1
```